UNIVERSIDAD DEL VALLE - SEDE TULUÁ

Facultad de Ingeniería

Ingeniería de Sistemas

ANAUJ, HERRAMIENTA CASE GENERADORA DE CÓDIGO FUENTE DE APLICACIONES WEB PROTOTIPO

TRABAJO DE GRADO PRESENTADO POR

HÉCTOR MAURICIO CABRERA MURILLO MEJÍA
201356599
hector.cabrera@correounivalle.edu.co

DIRECTOR
CARLOS ANDRÉS DELGADO SAAVEDRA. ING.

CODIRECTOR
VÍCTOR ANDRÉS BUCHELI GUERRERO. Ph.D.

Tuluá, Valle del Cauca
Mayo de 2016

Trabajo de grado presentado por
Héctor Mauricio Cabrera Murillo Mejía
Como requisito parcial para la obtención del título de Ingeniero de Sistemas

Carlos Andrés Delgado Saavedra, Ing.
Director

Víctor Andrés Bucheli Guerrero, Ph.D
Codirector

Oscar Orlando Ceballos Argote, Ms.C.
Jurado

A mi familia.

Índice general

Índice de figuras

Índice de tablas

Resumen

En este trabajo de grado se describe el proceso involucrado en el desarrollo de una herramienta CASE generadora de código fuente de aplicaciones Web prototipo a través de la programación funcional. Mediante las bases metodológicas relativas al desarrollo de software y sustentado en el paradigma funcional, se materializa un proceso con el que se construye y describe todos los aspectos requeridos en el desarrollo de la herramienta propuesta.

El desarrollo del proyecto es un esfuerzo por aprovechar los mecanismos y funciones especializadas permitidas por el paradigma funcional y que existen en el lenguaje de programación Racket, estableciendo una experiencia que sirva de referencia o iniciativa en la generación de aplicaciones Web prototipo soportadas en un esquema de base de datos relacional.

Introducción

Este trabajo de grado expone una posible solución a uno de los problemas que aqueja al proceso de desarrollo de software. Problema que puede catalogarse globalmente como "Fallas de implementación", debe su presencia a subproblemas de diferente origen y/o naturaleza, que en este caso son principalmente los que surgen en la fase de codificación. Esta fase como parte dominante del proceso, adquiere mayor complejidad mientras se avanza, involucra esfuerzo, tiempo y habilidades especificas. Es también sensible y con altas posibilidades de cometerse errores, capaces incluso de culminar el proceso, debido al alto grado de dependencia.

En especial atención a estas circunstancias donde el trabajo de grado cobra real importancia, expone a través de la programación funcional y mecanismos especializados, un medio alternativo que busca la reducción de tiempo involucrado y el surgimiento de posibles errores en la fase de codificación, al ser este un proceso configurable y automatizado.

A continuación se da a conocer lo que tratará cada capítulo.

En el capítulo *Contexto y objetivos*, se describe el problema percibido en el proceso de desarrollo de software, se establecen los objetivos, justifica y describe el alcance del proyecto. Seguido a esto, se presenta en el capítulo *Marco de referencia*, los temas relacionados al proyecto, se definen algunos conceptos y acrónimos utilizados más adelante en el documento y se describe los antecedentes tecnológicos.

A partir del capítulo *Análisis*, en adelante, se inicia la descripción correspondiente del proyecto, aquí se trata el análisis realizado, se describe los mecanismos utilizados en la generación de código fuente, se especifican los requerimientos funcionales y no funcionales, como estos fueron concebidos y evolucionaron en el proyecto.

En el capítulo *Diseño y arquitectura*, se considera la arquitectura y división modular de la herramienta, especificación de las estructuras de datos y almacenamiento, creación y relación de las funciones que cumplen los requerimientos y finalmente la elaboración del modelo relacional que soportará la herramienta.

Una vez considerados los requerimientos, su disposición y noción de funcionamiento, se inicia la implementación de la herramienta en el lenguaje de programación Racket. En el capítulo *Implementación*, se expone precisamente algunos aspectos importantes de la implementacion, teniendo en cuenta las directrices de diseño tratados en el capítulo previo.

Posteriormente en el capítulo *Pruebas*, se describen las pruebas realizadas en la herramienta, de aceptación, usabilidad y comparación, y se exponen los resultados generados. Finalmente en el capítulo *Conclusiones y trabajos futuros*, se dan a conocer las conclusiones finales y describen los trabajos futuros necesarios para resolver las limitaciones y perfeccionar la herramienta.

Capítulo 1

Contexto y Objetivos

1.1. Problema

1.1.1. Planteamiento del problema

El desarrollo de software es el proceso de tomar un grupo de requerimientos del usuario (un planteamiento del problema), analizarlo, diseñar una solución a el problema, implementar esa solución en un computador y finalmente probar su funcionalidad y desempeño. Este proceso involucra la programación, que es realmente la parte de implementación, o posiblemente la parte de diseño e implementación del desarrollo de software [1]. Como solución, lo que finalmente se obtiene es el software o aplicación, un programa informático capaz de realizar uno o varios tipos de tareas.

Existen múltiples formas de crear aplicaciones, bien sea de escritorio, Web o móvil. Se pueden emplear lenguajes de programación como COBOL, C, C++, Java, etc. [2]. Si se requiere que la aplicación sea soportada por la Web, es indispensable hacer uso de lenguajes de programación para tal fin como Java, PHP, Python, JavaScript, Perl, etc. Y, en adición a ellos, lenguajes de marcado como HTML, XML y CSS. De acuerdo a [2] la programación con estos lenguajes proporciona flexibilidad, control y rendimiento, pero se necesita mucho trabajo para aprovechar esa flexibilidad, y más trabajo aún para conseguir que la aplicación soporte un crecimiento continuado de su uso. Es bastante fácil cometer errores, o programar de forma que la aplicación falle o vaya más lento, en lugar de ir más rápido [3]. Se debe además ser consciente de que todo este trabajo está fijado a un límite de tiempo, condicionado por el uso de tecnologías, limitado por la capacidad humana y depende de lineamientos que normalmente cambian, propiciando la aparición de problemas (tareas imprevistas, mala planificación, recurso humano inadecuado, etc.), que fácilmente podrían conducir a sobrecostos, un producto final incompleto, o incluso la cancelación del proyecto.

Los inconvenientes asociados al desarrollo de software, algunos de ellos mencionados en [3] han estado siempre presentes en el ejercicio de esta actividad. Para mejorar esta situación fue necesario comprender mucho mejor las actividades involucradas en el desarrollo de software, crear métodos y herramientas que redujeran el esfuerzo requerido para producir sistemas grandes y complejos [4]. Estas últimas surgieron en los años 80, denominadas herramientas CASE *(Computer Aided Software Engineering)*[1].

[1] Ingeniería de Software Asistida por Computadora

"Las herramientas CASE proponen una nueva filosofía del concepto de ciclo de vida basado en la automatización, para lo cual proporcionan un conjunto de herramientas bien integradas, que, enmarcadas dentro de una determinada metodología, permiten automatizar las fases del ciclo de vida de un sistema de software. Con esta automatización total, se intenta conseguir una mejora en la productividad y en la calidad del producto final" [5].

Las fases a las que se hace referencia, son las que se han mencionado anteriormente análisis, diseño, codificación y pruebas, estas dos últimas definidas como implementación. El enfoque de ahora en adelante es en aquellas que automatizan un tipo de tarea del ciclo de vida del software, en este caso, las generadoras de código fuente en la fase de codificación.

Actualmente, existen múltiples herramientas CASE generadoras de código fuente, un gran porcentaje de ellas son propietarias y restringidas a ser usadas en sistemas operativos y software acompañante igualmente propietarios (CodeBhagat[2], CodeOnTime[3], CodeCharge[4], ASPRunner.NET[5], etc.), dejando un pequeño porcentaje restante como herramientas libres entre ellas (NoORM[6], CodeCooker[7], Celerio[8], etc.), algunas basando su generación de código a partir de un esquema de base de datos, sin evidencia de que alguna de ellas, con base a los resultados de un estudio previo, genere código fuente en el lenguaje de programación Racket.

"Racket, además de ser un lenguaje de programación, es un laboratorio de lenguajes de programación. Eso es, Racket viene con una única colección de mecanismos lingüísticos que permiten la construcción rápida de lenguajes confiables, fragmentos de lenguaje, y su composición" [6].

Esto a diferencia de los lenguajes de programación tradicionales, para este caso, es de gran ayuda, en la medida en que no se está obligado a adaptar el problema a un lenguaje, sino, adaptar el lenguaje al problema, reduciendo el esfuerzo involucrado en el desarrollo [7].

En este trabajo de grado, se propone una solución prototipo al problema de desarrollo de software en su fase de codificación, a través de la programación funcional, la construcción de un software CASE capaz de desplegar una aplicación Web prototipo mediante la generación de código fuente basado en un esquema de base de datos relacional.

Finalmente la presentación de una solución de software generador de código fuente basado en este lenguaje, es una oportunidad para demostrar sus capacidades y alcance.

[2]Para mayor información consultar en: https://www.codebhagat.com/
[3]Para mayor información consultar en: https://codeontime.com/
[4]Para mayor información consultar en: https://www.yessoftware.com/index2.php
[5]Para mayor información consultar en: http://xlinesoft.com/asprunnernet/index.htm
[6]Para mayor información consultar en: http://www.noorm.org/
[7]Para mayor información consultar en: http://codecooker.net/
[8]Para mayor información consultar en: http://www.jaxio.com/en/celerio.html

1.1.2. Formulación del problema

Lo que se busca con esta propuesta es determinar si a través de la programación funcional y mecanismos especializados que esta ofrece, es viable la construcción de una herramienta CASE enfocada en la generación de código fuente. Con el objetivo de dar respuesta a esta problemática se busca responder los siguientes interrogantes:

1. ¿Cómo crear una herramienta de software, a través de la programación funcional, capaz de crear un software prototipo preliminar basado en un esquema de base de datos previamente validado?
2. ¿Cómo permitir al software prototipo creado, tener una interfaz gráfica, que presente los datos y accione los mecanismos de persistencia?
3. ¿Cómo permitir al software prototipo creado, basado en un esquema de base de datos manipular su información con operaciones CRUD, y así ser persistente?
4. ¿Cómo evaluar el software prototipo creado y con ello validar su funcionalidad?

1.2. Objetivos

1.2.1. Objetivo General

Implementar una herramienta CASE de bajo nivel que permita generar código fuente de una aplicación Web prototipo, soportada en un esquema de base de datos relacional, utilizando el lenguaje de programación Racket.

1.2.2. Objetivos Específicos

1. Identificar qué elementos, procedimientos lingüísticos y funciones del lenguaje de programación Racket, permiten la creación de una herramienta CASE generadora de código fuente de una aplicación Web prototipo.
2. Analizar los requerimientos necesarios que permitan soportar el proceso de creación de una aplicación Web prototipo a partir de la generación de código fuente.
3. Desarrollar los módulos que permitan la creación, configuración y disposición de los elementos que conforman el esquema de base de datos del software prototipo.
4. Desarrollar los módulos que permitan la creación y activación de los mecanismos de persistencia provistos para los elementos que conforman el esquema de base de datos del software prototipo.

1.2.3. Resultados esperados

Los resultados que se esperan por cada objetivo específico se describen en la siguiente tabla:

Objetivo específico	Resultado	Capítulo/sección
1	Descripción de los mecanismos y elementos del lenguaje de programación Racket, dispuestos para la creación de un procedimiento capaz de generar código fuente.	Capítulo 3
2	Especificación de los requerimientos funcionales y no funcionales, que constituyen la base, para la creación de una herramienta CASE generadora de código fuente de aplicaciones Web prototipo.	Capítulo 3
3	Especificación de la organización física y diseño modular de los componentes de la herramienta CASE. Descripción de las estructuras de datos y archivos de almacenamiento. Descripción de los módulos, funciones y relación entre funciones, asociadas con la manipulación de los elementos del esquema de base de datos. Implementación de las funciones encargadas de recolectar, configurar y almacenar los elementos del esquema de base de datos. Validación de funciones responsables de manipular los elementos de la base de datos a través de pruebas de aceptación.	Capítulos 4, 5 y 6
4	Descripción de los módulos, funciones y relación entre funciones, asociadas con la manipulación de los mecanismos de persistencia. Implementación de las funciones responsables de la recolección, configuración y almacenamiento de los mecanismos de persistencia. Implementación de las funciones de generación de código fuente. Validación de funciones responsables de manipular los mecanismos de persistencia y de generación de código a través de pruebas de aceptación. Determinación de la idoneidad de la herramienta CASE a través de pruebas de usabilidad y comparación.	Capítulos 4, 5 y 6

Tabla 1.1: Resultados esperados

1.3. Justificación

1.3.1. Justificación Metodológica

Las herramientas CASE generadoras de código fuente son por naturaleza complejas de desarrollar, más aún cuando para ello se utilizan lenguajes de programación tradicionales que no están específicamente diseñados para abordar dicha tarea, requiriendo mucho más esfuerzo para alcanzar el objetivo propuesto. El lenguaje de programación Racket basado en el procesamiento de listas y la disponibilidad de mecanismos lingüísticos (funciones y procedimientos especializados para la creación y/o modificación de nueva sintaxis Ej: macros) permiten reducir el esfuerzo inherente en el desarrollo de este tipo de herramientas generadoras de código. Esto debido principalmente a la capacidad de generar código usando la expresividad inherente del lenguaje directamente, evitando en lo posible la mera concatenación de cadenas, un rasgo limitante y poco agradable a la vista.

Es por esto que se propone como objetivo, el desarrollo de una herramienta CASE generadora de código fuente, utilizando los mecanismos lingüísticos apropiados, principalmente macros, no presentes en los

lenguajes de programación tradicionales, acortando el esfuerzo necesario para desarrollar una herramienta de esta naturaleza.

1.3.2. Justificación Práctica

Organizaciones comerciales sin distinción en su tipo de actividad y tamaño, han basado completamente o parte de su operación en el uso de equipos de computo y consecuentemente el uso de software. Cuando las exigencias de estas operaciones, no pueden ser solventadas por el software existente, es necesario considerar su creación, tercerización o la compra de un nuevo software. Si se piensa en la creación, inmediatamente surge la pregunta, ¿Cómo hacerlo?, cuya respuesta hace referencia al proceso de desarrollo de software. Normalmente en este proceso, luego de un análisis y diseño, la fase de codificación inicia desde cero.

Desarrollar una herramienta CASE generadora de código fuente a través de los mecanismos lingüísticos de Racket, centra el esfuerzo en conseguir no una aplicación tradicional, sino un lenguaje apropiado, extensible, capaz de alterar o adquirir más definiciones en su sintaxis, que puedan cubrir nuevas necesidades, algo difícil de lograr con los lenguajes tradicionales.

Finalmente, enfocar la solución como un lenguaje para la generación de código fuente, fomenta el desarrollo de este tipo de herramientas y/o de necesidades similares en dicha forma e impulsa el uso de lenguajes principalmente funcionales como Racket.

1.4. Alcance del proyecto

Anauj es una herramienta CASE generadora de código fuente de aplicaciones Web prototipo que estén soportadas en un esquema de base de datos relacional. No se pretende generar una aplicación completa que pueda ponerse en producción de forma inmediata, por el contrario, la herramienta está limitada a crear los elementos gráficos asociados al esquema de base de datos y los mecanismos de persistencia (CRUD).

Para una mejor especificación del alcance del proyecto, se detalla el resultado esperado por cada objetivo.

1. Identificar que elementos, procedimientos lingüísticos y funciones del lenguaje de programación Racket permiten la creación de una herramienta CASE generadora de código fuente de una aplicación Web prototipo.

    - El alcance de este objetivo está fijado a la búsqueda, estudio y conocimiento de los procedimientos del lenguaje de programación Racket. No se pretende elaborar un documento formal por escrito de los procedimientos estudiados, solo habrá evidencia de ellos en la implementación del generador.

2. Analizar los requerimientos necesarios que permitan soportar el proceso de creación de una aplicación Web prototipo a partir de la generación de código fuente.

    - El alcance de este objetivo está limitado al cumplimiento de los siguientes requerimientos: registro y autenticación del usuario para el uso de la herramienta, selección y visualización del esquema de base de datos. La visualización del esquema (tablas y campos) se hace en forma textual y no gráfica. Extracción del código fuente de la aplicación Web prototipo parcial (por tabla) o completa.

3. Desarollar los módulos que permitan la creación, configuración y disposición de los elementos que conforman el esquema de base de datos del software prototipo.

    - El alcance de este objetivo involucra: selección, configuración y visualización de los campos de cada tabla del esquema de base de datos. En la configuración de cada campo está permitido: editar el nombre de la etiqueta que lo acompaña, elegir el tipo de elemento que representa al campo, el tipo de dato (restricción), tamaño y establecimiento de valores por defecto para elementos select, checkbox y radio. La disposición de los elementos está limitada a ser horizontal o vertical. La configuración de cada campo la hace el usuario a través de una tabla dinámica y su visualización es generada cuando lo desee, es decir el usuario puede alternar entre la configuración y la visualización de esa configuración.

4. Desarrollar los módulos que permitan la creación y activación de los mecanismos de persistencia provistos para los elementos que conforman el esquema de base de datos del software prototipo.

    - El alcance de este objetivo involucra: creación, selección y activación de los mecanismos de persistencia CRUD (crear, leer, actualizar y eliminar). El usuario puede seleccionar los mecanismos necesarios para su aplicación Web prototipo y editar limitadamente su funcionamiento. Esta edición obedece únicamente a fijar los campos que intervienen en los mecanismos.

1.5. Estructura del documento

El contenido del documento se encuentra estructurado de la siguiente manera:

En el capítulo *Marco de referencia*, se describen los temas relacionados con el proyecto, se definen algunos conceptos y acrónimos que serán utilizados a lo largo del documento y se adentrará en lo concerniente a las herramientas CASE generadoras de código fuente y su estado actual. Posteriormente, debido a que en el capítulo *Contexto y objetivos*, se han determinado los objetivos y el alcance del proyecto, se inicia desde aquí, su ejecución, considerandose en primera instancia los mecanismos que permiten la generación de código fuente y descripción de requerimientos, contenidos en el capítulo de *Análisis*, que comprende los objetivos específicos 1 y 2. Después de conocer lo requerido en el proyecto, es necesario establecer como será construido, cuales serán sus componentes, formas de acción, disposición y ordenamiento. Esto se detalla en el capítulo *Diseño y arquitectura*, que comprende los objetivos específicos 3 y 4. En el capítulo siguiente, *Implementación*, se presenta parcialmente algunas de las piezas de implementación más relevantes logradas en el proyecto, abarcando parte de los objetivos específicos 3 y 4.

Posteriormente en el capítulo *Pruebas*, se dan a conocer las pruebas efectuadas que evalúan la ejecución correcta, usabilidad y funcionalidad del proyecto. Por último, en el capítulo *Conclusiones y trabajos futuros*, se manifiestan las conclusiones finales y describen los trabajos necesarios para resolver las limitaciones y perfeccionar la herramienta obtenida.

Capítulo 2

Marco de referencia

En este capítulo se definen algunos de los conceptos más importantes que contextualizan y facilitan el entendimiento del problema y los pasos seguidos para su solución.

En primera instancia se define en forma general los principales conceptos con los que se relaciona el proyecto, posteriormente se define cada término y acrónimo referido en el desarrollo del proyecto.

2.1. Marco teórico

2.1.1. Herramientas CASE

Las herramientas CASE proponen una nueva filosofía del concepto de ciclo de vida basado en la automatización, para lo cual proporcionan un conjunto de herramientas bien integradas, que, enmarcadas dentro de una determinada metodología, permiten automatizar las fases del ciclo de vida de un sistema de software. Con esta automatización total, se intenta conseguir una mejora en la productividad y en la calidad del producto final [5].

Las herramientas CASE, en función de las fases del ciclo de vida del desarrollo del software, se pueden agrupar de la siguiente forma: [8]

- **Herramientas integradas, I-CASE (Integrated CASE):** Abarcan todas las fases del ciclo de vida del desarrollo de sistemas, son llamadas también CASE workbench.
- **Herramientas de alto nivel, U-CASE (Upper CASE):** También denominadas front-end, orientadas a la automatización y soporte de actividades desarrolladas durante las primeras fases del ciclo de vida del desarrollo de software: análisis y diseño.
- **Herramientas de bajo nivel, L-CASE (Lower CASE):** También denominadas back-end, dirigidas a las últimas fases del ciclo de vida del desarrollo de software: desarrollo e implantación.
- **Juego de herramientas o tools-CASE:** Son el tipo más simple de herramientas CASE. Automatizan una fase del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

2.1.2. Herramientas CASE generadoras de código fuente

Las herramientas CASE generadoras de código fuente representan una fracción de un conjunto más amplio de herramientas y metodologías que abarcan todas las fases del ciclo de vida del desarrollo de software. Su principal objetivo es automatizar la codificación del software. Como su nombre lo indica generan código fuente en un lenguaje de programación específico y basan su generación en información proveniente de la fase de diseño (Diagramas UML, esquema de base de datos, etc.) [4].

Las características más importantes de las herramientas generadoras de código fuente son: [9]

- Lenguaje generado. Si se trata de un lenguaje estándar o un lenguaje propietario.
- Portabilidad del código generado. Capacidad para poder ejecutarlo en diferentes plataformas físicas y/o lógicas.
- Generación del esqueleto del programa o del programa completo. Si únicamente genera el esqueleto será necesario completar el resto mediante programación.
- Posibilidad de modificación del código generado. Suele ser necesario acceder directamente al código generado para optimizarlo o completarlo.
- Generación del código asociado a las pantallas e informes de la aplicación. Mediante esta característica se obtiene la interfaz de usuario de la aplicación.

2.1.3. Modelo entidad relación

El modelo de datos entidad-relación (E-R) está basado en una percepción del mundo real consistente en objetos básicos llamados entidades y de relaciones entre estos objetos. Se desarrolló para facilitar el diseño de base de datos permitiendo la especificación de un esquema de la empresa que representa la estructura lógica completa de una base de datos [10].

Hay tres nociones básicas que emplea el modelo de datos E-R: conjuntos de entidades, conjuntos de relaciones y atributos.

- **Conjuntos de entidades**

  Una **entidad** es una cosa u objeto en el mundo real que es distinguible de todos los demás objetos. Una entidad puede ser concreta, como una persona o un libro, o puede ser abstracta, como un préstamo, unas vacaciones o un concepto.

  Un **conjunto de entidades** es un conjunto de entidades del mismo tipo que comparten las mismas propiedades, o atributos. El conjunto de todas las personas que son clientes en un banco dado, por ejemplo, se pueden definir como el conjunto de entidades cliente.

  Una entidad se representa mediante un **conjunto de atributos**. Los atributos describen propiedades que posee cada miembro de un conjunto de entidades. Posibles atributos del conjunto de entidades cliente son id-cliente, nombre-cliente, calle-cliente y cuidad-cliente. Cada entidad tiene un valor para cada uno de sus atributos.

Para cada atributo hay un conjunto de valores permitidos, llamados **dominio**, o el **conjunto de valores** de ese atributo. El dominio del atributo nombre-cliente podría ser el conjunto de todas las cadenas de texto de cierta longitud.

Un atributo, como se usa en el modelo E-R, se puede caracterizar por los siguientes tipos de atributo. Los **atributos simples**, aquellos que no están divididos en subpartes, es decir, no se pueden subdividir en otros atributos. Los **atributos complejos**, aquellos que pueden subdividirse en subpartes, es decir, en otros atributos. Por ejemplo, nombre-cliente se puede componer de nombre, primer-apellido y segundo-apellido. Los **atributos monovalorados** hacen referencia a un único valor. Por ejemplo, número-préstamo hace referencia a un único número de préstamo. Los **atributos multivalorados** hacen referencia a un conjunto de valores. Por ejemplo, número-teléfono puede hacer referencia a 0, 1 o más números de teléfonos. Los **atributos derivados** son aquellos cuyo valor se deriva de valores de otros atributos o entidades relacionadas.

- **Conjunto de relaciones**

  Una **relación** es una asociación entre diferentes entidades. Por ejemplo, se puede definir una relación que asocie al cliente López con el préstamo P-15. Esta relación especifica que López es un cliente con el préstamo número P-15. Un **conjunto de relaciones** es un conjunto de relaciones del mismo tipo.

La estructura lógica general de una base de datos se puede expresar gráficamente mediante un diagrama E-R, que consta de los siguientes componentes:

- **Rectágulos**, que representan conjuntos de entidades.
- **Elipses**, que representan atributos.
- **Rombos**, que representan relaciones entre conjuntos de entidades.
- **Líneas**, que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.

Como ilustración, considérese parte de una base de datos de un sistema bancario consistente en clientes y cuentas que tienen esos clientes. En la siguiente figura se muestra el diagrama E-R correspondiente [10].

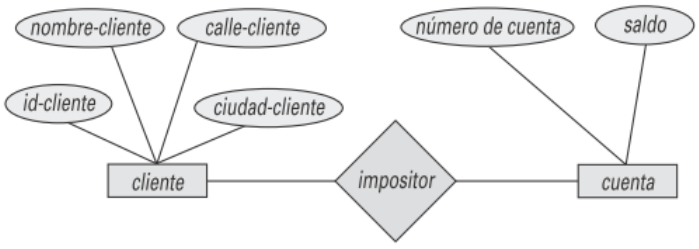


Figura 2.1: Diagrama de modelo entidad relación [10]

Un esquema E-R puede definir ciertas restricciones a las que los contenidos de la base de datos se deben adaptar. Las restricciones de mayor relevancia son **la correspondencia de cardinalidades** y **restricciones de participación**.

La **correspondencia de cardinalidades** expresa el número de entidades a las que otra entidad puede estar asociada vía un conjunto de relaciones. Por ejemplo para un conjunto de relaciones binarias R entre dos conjuntos de entidades A y B, la correspondencia de cardinalidades debe ser: [10]

- **Uno a uno**. Una entidad en A se asocia con a lo sumo una entidad en B, y una entidad en B se asocia con a lo sumo una entidad en A.
- **Uno a varios**. Una entidad en A se asocia con cualquier número de entidades en B (ninguna o varias). Sin embargo, una entidad en B, se puede asociar con a lo sumo una entidad en A.
- **Varios a uno**. Una entidad en A se asocia con a lo sumo una entidad en B. Sin embargo, una entidad en B, se puede asociar con cualquier número de entidades en A (ninguna o varias).
- **Varios a varios**. Una entidad en A se asocia con cualquier número de entidades en B, y una entidad en B se asocia con cualquier número de entidades en A.

Las **restricciones de participación** definen el tipo de participación de un conjunto de entidades en un conjunto de relaciones. Los tipos son total o parcial [10].

- **Total**. La participación es total si cada entidad en E participa al menos en una relación en R. Se representan en el diagrama E-R con una línea de doble trazo.
- **Parcial**. La participación es parcial si sólo algunas entidades en E participan en relaciones en R. Se representan en el diagrama E-R con una línea de un solo trazo.

Es necesario además tener una forma de distinguir cada entidad y relación en un conjunto de entidades y relaciones. Esta diferencia entre ellas se expresa en término de sus atributos. Es decir, los atributos de una entidad deben ser suficientes para identificar unívocamente a la entidad.

Esta distinción se logra a través de la **clave primaria**. Un conjunto de uno o más atributos que identificará de forma única una entidad dentro de un conjunto de entidades, y ningún subconjunto de ella tendrá esta propiedad. Se representa en el diagrama E-R con los atributos subrayados [10].

Como ilustración considérese la relación entre cliente y préstamo, cuyo diagrama E-R se muestra en la siguiente figura.

Figura 2.2: Diagrama de modelo E-R para relación entre cliente y préstamo [10]

La cardinalidad 1..1 entre *préstamo* y *prestamista* significa que cada préstamo debe tener exactamente un cliente asociado. El limite 0..* entre *cliente* y *prestamista* indica que un cliente puede tener ninguno o varios préstamos. Por consiguiente la relación *prestamista* es 1 a varios de *cliente* a *préstamo*, y la participación de *préstamo* en *prestamista* es total.

2.1.4. Modelo relacional

El modelo relacional es usado para expresar el diseño de una base de datos. Se basa en el concepto de relación, que se representa físicamente como una tabla o arreglo bidimensional. En este modelo, las filas corresponden a registros individuales o tuplas y las columnas corresponden a atributos [11].

Por ejemplo considérese la tabla *cuenta*, en ella hay tres columnas: número-cuenta, nombre-sucursal y saldo. En el modelo relacional estas columnas hacen referencia a los **atributos**. Para cada atributo existe un conjunto de valores permitidos, denominado **dominio** de ese atributo. Si se considera a D1, D2 y D3 como conjuntos dominio de los atributos número-cuenta, nombre-sucursal y saldo respectivamente, se puede decir que la tabla *cuenta* contendrá un subconjunto del conjunto de todas las filas posibles, y las filas deberán consistir en una **tupla** (v1, v2, v3) donde v1 pertenece al dominio D1, v2 pertenece al dominio D2 y v3 pertenece al dominio D3 [10].

Al momento de especificar el modelo relacional es necesario tener presente los siguientes conceptos [11].

- **Clave primaria**: Conjunto mínimo de atributos de una relación que se elige para identificar de manera única cada tupla.
- **Clave externa**: Es un atributo o combinación de atributos de una relación que no es la clave primaria de dicha relación, pero es clave primaria de otra relación.
- **Integridad de entidad**: Establece que ningún atributo de una clave primaria puede tener un valor nulo.
- **Integridad referencial**: Si en una relación existe una clave externa, su valor debe coincidir con el valor de la clave primaria de alguna tupla de su relación base, o debe ser completamente nulo.

A partir del modelo E-R se puede llegar al modelo relacional y desde este ser implementado en un sistema de gestión de base de datos. Por ejemplo el modelo relacional para el modelo E-R de la Figura 2.2 es:

CLIENTE(id-cliente, nombre-cliente, calle-cliente, ciudad-cliente)
PRÉSTAMO(número-préstamo, id-cliente, importe)
Clave externa: id-cliente - Hace referencia a CLIENTE.

2.1.5. Sistema de gestión de base de datos

Un sistema de gestión de base de datos (DBMS, Database Management System) es software informático que gestiona el acceso, almacenamiento, modificación y extracción de información de una base de datos.

Un DBMS de multiusuario típico realiza las tareas siguientes, y más: [12]

- Un DBMS gestiona con seguridad el acceso compartido a una base de datos entre varios usuarios simultáneos.
- Un DBMS utiliza en forma adecuada los recursos de la computadora, de modo que un gran número de usuarios de aplicaciones pueden trabajar con tiempos de respuesta cortos para lograr una productividad máxima.
- Un DBMS protege la información de la base de datos, de tal manera que pueda reconstruir el trabajo perdido debido a cualquier contingencia, desde un simple corte en el fluido eléctrico hasta un desastre catastrófico en la ubicación.

2.1.6. SQL

El lenguaje SQL (Structured Query Language) o lenguaje de consulta estructurado, es un lenguaje de consulta y de gestión de bases de datos relacionales [13].
El SQL es a la vez un Lenguaje de Manipulación de Datos (DML: Data Manipulation Language) y un Lenguaje de Definición de Datos (DDL: Data Definition Language). Su parte DDL permite crear y modificar la estructura básica de una base de datos tanto a nivel físico como lógico. La parte DML se emplea para la manipulación de los contenidos de la base de datos [14].
Administradores de bases de datos, desarrolladores y usuarios de aplicaciones lo usan para: [12]

- Recuperar, introducir, actualizar y eliminar los datos de bases de datos.
- Crear, cambiar y colocar los objetos de bases de datos.
- Restringir el acceso a los datos de bases de datos y a las operaciones del sistema.

2.1.7. Generación de código fuente

La generación de código es la técnica de construir y usar programas para escribir otros programas. Un ejemplo concreto es usar un generador de código para crear código de acceso a una base de datos o capas de procedimiento remotas. La idea central es crear código consistente y de alta calidad más rápido [15].

Dentro de la generación de código activa son muchos los tipos de modelos que abarcan un rango de soluciones desde muy simples y pequeñas a grandes y complejas. Son muchas las formas para categorizar a los generadores.

En esta sección que sigue se examinan seis modelos: [15]

- **Code munging (Código munging)**

  Munging es la jerga para retorcer y formar algo de una forma a otra. Dado un código de entrada, el munger elige rasgos importantes y los usa para crear uno o varios archivos de salida de varios tipos.

- **The inline-code expander (Expandidor de código en línea)**

  Un expandidor de código en línea toma código fuente como entrada y crea código de producción como salida. El archivo de entrada contiene marcas especiales que el expandidor reemplaza con código de producción cuando este crea el código de salida.

- **Mixed-code generation (Generación de código mixto)**

  Un generador de código mixto lee un archivo de código fuente y luego modifica y reemplaza el archivo en su lugar. Este es diferente del expandidor de código en línea debido a que el generador de código mixto pone la salida del generador dentro del archivo de entrada.

- **Partial-class generation (Generación de clases parciales)**

  Un generador de clases parciales lee un archivo de definición abstracto que contiene suficiente información para construir un grupo de clases. Luego, este usa plantillas para construir librerias de clases base. Estas clases son compiladas con clases construidas por los ingenieros para completar la producción del grupo de clases.

- **Tier o layer generation (Generación de capas)**

  En este modelo, el generador toma responsabilidad para construir una capa completa de un sistema de n-capas.

- **Full-domain language (Lenguaje de dominio completo)**

  Un lenguaje de dominio completo es un lenguaje de Turing completo, modificado para permitir a los ingenieros representar conceptos en el dominio mas fácilmente. Un lenguaje de Turing completo es un lenguaje de computador de propósito general que soporta todo, desde manejo de variables, lógica, bifurcación, funcional, y capacidades de descomposición de objetos incluidas hoy en los lenguajes de programación.

2.1.8. Lenguaje de programación Racket

Racket es un lenguaje de programación de espectro completo. Este va más allá de Lisp y Scheme con dialectos que soportan objetos, tipos, lazy evaluation y más. Racket permite a los programadores enlazar componentes escritos en diferentes dialectos, y da la capacidad a los programadores para crear nuevos, dialectos específicos del proyecto. Las librerias Racket soportan aplicaciones desde servidores Web y bases de datos a interfaces gráficas de usuario y diagramas.

"Racket es a la vez un lenguaje de programación y un marco para construir lenguajes de programación. Un programa Racket puede contener definiciones que extienden la sintaxis del lenguaje para usar luego en el mismo programa, y pueden ser empaquetadas en módulos para uso en múltiples programas. Racket soporta un camino sin problemas desde extensiones del lenguaje relativamente simples a lenguajes completamente nuevos, ya que una herramienta de programación, como cualquier otra pieza de software, es posible empezar de forma sencilla y crecer tanto como las demandas en el lenguaje incrementen" [16].

2.1.9. Aplicaciones Web

Una aplicación Web es una aplicación que está desarrollada para la Web. Esta permite a los usuarios llevar a cabo una tarea, como obtener información, comprar, aplicar para un trabajo, escuchar radio en internet o cualquiera de las muchas actividades posibles en la Web.

Las páginas Web son más complejas y/o útiles cuando están conectadas a una base de datos. Tales páginas son denominadas páginas manejadoras de base de datos [17].

Las aplicaciones Web son, por lo tanto, programas por computador que permiten a los visitantes del sitio Web enviar y recuperar datos a/desde una base de datos sobre Internet, usando el navegador Web preferido. Los datos son entonces presentados al usuario dentro de su navegador como información generada dinámicamente (en un formato específico, p. ej. HTML usando CSS) por la aplicación Web a través de un servidor Web.[1]

2.1.10. Arquitectura de una aplicación Web

La arquitectura software de un programa o sistema computacional es la estructura o estructuras del sistema, el cual comprende elementos de software, las propiedades externamente visibles de aquellos elementos, y las relaciones entre ellos [18]. Los elementos del software proveen la funcionalidad del sistema, mientras que los atributos de calidad requeridos (rendimiento, adaptabilidad, usabilidad, modificabilidad, etc.) son principalmente alcanzados a través de las estructuras de la arquitectura del software [18].

Durante el diseño arquitectónico, arquitectos aplican enfoques arquitectónicos probados para transferir los requerimientos del sistema dentro de una apropiada estructura de software. Los patrones arquitectónicos ofrecen soluciones bien establecidas a problemas de arquitectura. Un patrón arquitectónico expresa un esquema de organización estructural fundamental para un sistema de software el cual exhibe atributos de calidad conocidos. Por ejemplo, Las capas es un patrón bien conocido que estructura un sistema de software dentro de un apropiado número de capas que están ubicadas cada una en la parte superior de otra. Los servicios de cada capa implementan una estrategia para combinar los servicios de la capa de abajo en una forma significativa. Las capas mejoran la mantenibilidad, extensibilidad y reusabilidad del sistema [19].

Un modelo arquitectónico ampliamente utilizado en el desarrollo de aplicaciones de escritorio y aplicaciones Web es el de tres capas, constituido por las capas de presentación, lógica de negocios y persistencia. (Ver Figura 2.3)

La **capa de presentación** presenta el sistema al usuario, le muestra la información y captura la información del usuario.

La **capa de lógica de negocios** es donde se desarrollan los algoritmos propios de la aplicación. En esta capa se implementa la lógica obtenida por el análisis de requerimientos del proyecto. Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados y con la capa de persistencia, para solicitar la información pertinente al motor de base de datos.

La **capa de persistencia** es donde se almacenan los datos y se realiza las operaciones para acceder a los mismos. Esta capa recibe solicitudes de almacenamiento o recuperación de información desde la capa de la lógica del negocio [20].

[1] http://www.acunetix.com/websitesecurity/web-applications/ - Consultado: 26 de Febrero de 2016

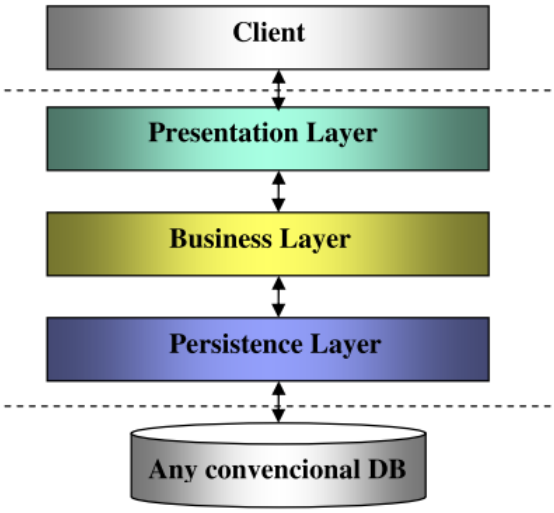


Figura 2.3: Modelo de arquitectura de 3 capas [21]

2.2. Marco Conceptual

Algunos conceptos y acrónimos referidos en el desarrollo del proyecto son:

Concepto	**Definición**
Abstracción	Operación intelectual que separa las cualidades de un objeto o cosa para considerarlas aisladamente. Como cuando se describe el cuerpo humano y hacemos referencia a cabeza, brazos, piernas, etc.
Artefacto	Es el resultado obtenido por la realización de una actividad específica en el proceso de desarrollo de software. El resultado es físico y evidenciable en un documento y/o diagrama. Sus diferentes tipos están definidos por UML bajo el proceso RUP.
Navegador	Es una herramienta que permite visualizar los contenidos de la web e interactuar con las aplicaciones web.
Capa	En desarrollo de software, es la división funcional de un sistema, modelo o arquitectura. El hacerlo permite concentrar una función específica en cada capa, facilitando el desarrollo y comprensión de este.
Cliente/Servidor	Arquitectura que divide la ejecución de procesos en una red entre una o varias computadoras que solicitan los servicios - cliente - y la computadora o computadoras que proporcionan los servicios - servidor.
Código de producción	Es aquel código fuente que conforma una aplicación que es funcional y puede ser utilizada en un entorno de producción por los usuarios finales, ajeno al entorno de desarrollo.
Correspondencia con patrón	Es una técnica para determinar la ocurrencia de un carácter o secuencia de caracteres particular en un texto, utilizando expresiones regulares.
Dialecto	En informática es la variación producida a partir de un lenguaje de programación principal. En esta variación se puede conservar y/o adicionar nuevas características al lenguaje.
Expresión regular	Expresión utilizada en correspondencia con patrones, esta consta de letras y números, además de caracteres especiales que tienen funciones específicas en la búsqueda, llamados metacaracteres.

Interfaz gráfica de usuario	Una interfaz de usuario utiliza objetos tales como iconos, paneles, menús, punteros, barras de desplazamiento, etc. Que permiten al usuario utilizar aplicaciones a través de interacciónes gráficas, como movimiento, selección, apuntación y pulsación.
Lenguaje de consulta	Lenguaje empleado por usuarios de bases de datos, que a través de instrucciones se recupera, modifica, añade y suprimen datos. Este conjunto principal de funciones se les denomina CRUD.
Librería	Conjunto de archivos que en su contenido tienen principalmente definidas variables y funciones.
Macros	Una macro es una forma sintáctica con un transformador asociado que expande la forma original dentro de formas existentes. En otras palabras, una macro es un mecanismo que permite extender el lenguaje de programación, en este caso Racket, incluyendo una nueva sintaxis.
Módulos	Vease Librería.
Sintaxis	Son el conjunto de reglas que definen la secuencia y disposición correcta de los elementos propios de un lenguaje de programación.
Templates	Template (Plantilla) es un modelo o patrón de documento que proporciona los elementos básicos para la creación de un tipo específico de documento.
WWW	World Wide Web, comúnmente conocida como web, es la red mundial de información distribuida que permite la transmisión de documentos de hipertexto e hipermedia.

Tabla 2.2: Tabla de conceptos

Acrónimo	**Definición**
CASE	Computer Aided Software Engineering
CRUD	Create Read Update Delete
UML	Unified Modeling Language
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
URL	Uniform Resource Locator
SQL	Structured Query Language

Tabla 2.3: Tabla de acrónimos

2.3. Estado del arte

2.3.1. Antecedentes Tecnológicos

A continuación se referencia algunas de las herramientas CASE generadoras de código fuente existentes en el mercado.

- **CodeBhagat v1.0 for NET - Enterprise Edition**

  CodeBhagat es un producto completo que incluye todas las características necesarias para generar código. Permite generar código directamente desde 8 bases de datos soportadas (SQL Server, Oracle, MySQL, SQL Azure, SQL Express, SQL Standalone Database files (.mdf), SQL CE/Mobile and MS-Access). Permite generar código para lenguajes C#.NET o VB.NET. Requiere de MS Windows XP

o una versión de sistema operativo más avanzada, aplicativo MS .NET Framework 3.5 o versión más avanzada.[2]

- **CodeOnTime Standard Edition**

  CodeOnTime es un producto capaz de crear aplicaciones Web avanzadas desde la base de datos. Aplicaciones ricas en características tales como, filtración adaptable, barras de búsqueda, reportes, gráficos, etc. Incluye completo acceso al código generado, soporta última versión de Microsoft .NET Framework 4.0, soporte para bases de datos (Microsoft SQL Server, Oracle, MySQL, IBM DB2, SAP SQL Anywhere, PostgreSQL).[3]

- **CodeCharge Studio 5.1**

  CodeCharge Studio es un producto que permite la creación de aplicaciones Web manejadoras de base de datos visualmente con un mínimo de cantidad de código. Incluye constructor de aplicaciones que convierte una base de datos (MS Access, MS SQL, MySQL, Oracle, etc.) en una aplicación Web funcional. Generación dinámica de sitios Web libre de errores para los lenguajes PHP, ASP, JSP, Perl, ColdFusion, ASP.NET. Disponibilidad de soluciones pre-construidas para modificar y realzar (Tienda online, sistema manejador de tareas, portal comunitario, etc.). Requiere de sistema operativo Windows XP, 2000, 2003, Vista, 7, 2008, 2012, 8, 8.1.[4]

- **CodeCooker**

  CodeCooker es un aplicativo Web que permite la creación de diagramas UML. Actualmente genera diagramas UML en versión 2.0 (Con la excepción de asociaciones). Se encuentra integrado con Google Drive, para guardar los diagramas. Permite generar código en los lenguajes C++, C#, CoffeScript y JSON a partir de los diagramas UML (modelos).[5]

- **ASPRunner.NET**

  ASPRunner.NET crea aplicaciones ASP.NET que permite al usuario buscar, editar, eliminar y agregar datos a bases de datos, tales como, Oracle, SQL Server, MS Access, PostgreSQL, o MySQL. Provee múltiples esquemas de distribución de elementos, colores para esquemas y editor visual con capacidades de arrastrar y soltar. Permite la creación de gráficos e informes personalizables para complementar el sitio Web. Ofrece mecanismos de registro para nuevos usuarios y protección contra usuarios no legítimos. Facilita la manipulación de las bases de datos a través del software, permitiendo la modificación de tablas y campos.[6]

[2] http://www.codebhagat.com/Store/ShowProduct.aspx?id=1 - Consultado: 26 de Febrero de 2016
[3] https://codeontime.com/standard-edition-features - Consultado: 16 de Octubre de 2014
[4] http://www.yessoftware.com/products/product.php?product_id=1 - Consultado: 26 de Febrero de 2016
[5] http://codecooker.net/ - Consultado: 26 de Febrero de 2016
[6] http://xlinesoft.com/asprunnernet/index.htm - Consultado: 26 de Febrero de 2016

Capítulo 3

Análisis

3.1. Análisis

La construcción de software exige el establecimiento de requerimientos funcionales que definen las acciones, comportamiento y capacidades de este y requerimientos no funcionales que imponen restricciones sobre su diseño e implementación.

En este capítulo se identifican los mecanismos necesarios para la generación de código fuente, los requerimientos funcionales y no funcionales que satisfacen los objetivos planteados, se describe en forma general como fueron obtenidos y que cambios tuvieron en el transcurso de su desarrollo.

3.1.1. Generación de código fuente

Uno de los componentes más importantes y motivadores del proyecto es la generación de código fuente, lo que se hará a continuación es describir los mecanismos que lo permiten y más adelante en el capítulo de implementación mostrar como fueron utilizados y adaptados al proyecto.

El mecanismo base soportado por el lenguaje Racket que permite la construcción de sintaxis son las macros. La macro es una forma sintáctica con un transformador asociado que expande la forma original en las formas existentes.[1] En términos simples es un transformador de sintaxis, una función que toma sintaxis y devuelve sintaxis, una sintaxis transformada.[2]

[1] Para mayor información consultar en: http://docs.racket-lang.org/guide/macros.html - Consultado: 26 de Febrero de 2016

[2] Para mayor información consultar en: http://www.greghendershott.com/fear-of-macros/ - Consultado: 26 de Febrero de 2016

Un ejemplo de un transformador es:

Código Fuente 3.1: Ejemplo básico de una macro

```
> (define-syntax foo
        (lambda (stx)
                (syntax "I␣am␣foo")))
> (foo)
"I␣am␣foo"
```

Que ignora su sintaxis de entrada y siempre retorna la sintaxis de una cadena.

La función syntax tiene una abreviatura que es #', y al agregarlo en el ejemplo base resultaría:

Código Fuente 3.2: Abreviatura de función syntax

```
> (define-syntax (quoted-foo stx)
        #'"I␣am␣also␣foo,␣using␣#'␣instead␣syntax")
> (quoted-foo)
"I␣am␣also␣foo,␣using␣#'␣instead␣syntax")
```

Alrededor de la función syntax existen funciones adicionales que permiten la manipulación de la sintaxis. Algunas son:

- **syntax→datum** convierte la sintaxis entrante en su S-expression (expresión Racket) correspondiente bajo una lista.
- **datum→syntax** convierte una S-expression (expresión Racket) en su sintaxis correspondiente.

```
; Define la sintaxis
(define stx #'(if x (list true) #f))
#<syntax:3:14 (if x (list true) #f)>

; Convierte sintaxis a expresión Racket
(define lstexp (syntax->datum stx))
'(if x (list true) #f)

; Convierte expresión Racket a sintaxis
(define stxagain (datum->syntax #f lstexp))
#<syntax (if x (list true) #f)>
```

Cuando se transforma sintaxis generalmente lo que se hace es tomar las piezas, otorgar algún orden, agregar y/o modificar algunas piezas. Un ejemplo más completo sería invertir el orden de la sintaxis entrante.

Código Fuente 3.3: Invertir sintaxis

```
> (define-syntax (reverse-me stx)
    (datum->syntax stx (reverse (cdr (syntax->datum stx)))))
> (reverse-me "backwards" "am" "i" values)
"i"
"am"
"backwards"
```

Una vez se recibe la sintaxis, con syntax → datum se obtiene una lista de las expresiones de esa sintaxis, es decir

```
'(reverse-me "backwards" "am" "i" values)
```

Se remueve el primer argumento *reverse-me* con la función *cdr* resultando en

```
'("backwards" "am" "i" values)
```

Seguido a eso se invierte la lista a través de la función *reverse* resultando en

```
'(values "i" "am" "backwards")
```

Finalmente con datum→syntax se retorna la sintaxis

```
#'<syntax (values "i" "am" "backwards")>
```

Que es lo que el compilador de Racket recibe y evalúa

```
> (values "i" "am" "backwards")
```

Una característica importante a tener en cuenta en la definición de nueva sintaxis, es controlar el orden y cantidad de las piezas que la definen. Esto se consigue a través de la función *syntax-case*. Por ejemplo para la definición de un nuevo *if* se haría lo siguiente.

```
> (define-syntax (our-if stx)
    (syntax-case stx ()
      [(_ condition true-expr false-expr)
       #'(cond [condition true-expr]
               [else false-expr])
                         ]))
> (our-if #t "true" "false")
"true"
> (our-if #t "true") ; error
> (our-if #t "true" "false" "another-thing") ; error
```

En este caso *our-if* es el nombre de la nueva sintaxis que se corresponde con el caracter _, el cuerpo o forma de la sintaxis es *(_ condition true-expr false-expr)*, incluyendo el nombre, obliga que siempre en su utilización se tenga una condición, la respuesta si la condición se cumple y si no. En la utilización de este nuevo *if* si se agrega un nuevo argumento o se ignora alguno, se obtiene un error. La construcción *[(old-syntax) (new-syntax)]* es la que hace corresponder la sintaxis entrante *(old-syntax)* con la sintaxis nueva o transformada *(new-syntax)*. La sintaxis transformada que se devuelve en este ejemplo es la asociada a la estructura de control *if* a través de función *cond*. Recuerde que #' devuelve sintaxis.

3.1.2. Requerimientos funcionales

En la descripción de cada requerimiento funcional se tiene en cuenta la siguiente información:

- **Identificador**, asignación que identifica de forma única a cada requerimiento.
- **Nombre del requerimiento**, nombre que identifica y relaciona el requerimiento con su función.
- **Actor(es)**, rol o roles que interactúan con el requerimiento.
- **Indispensable/Deseable**, clasificación del requerimiento según su relevancia.

- **Prioridad**, asignación que otorga orden al desarrollo del requerimiento. Se clasifica en alta, media o baja.
- **Visible/No visible**, especifica si requerimiento es visible o no al usuario final.
- **Autor**, persona encargada de implementar el requerimiento.
- **Fecha de elaboración**, fecha en que se concluye implementación del requerimiento.
- **Revisión**, persona encargada de corroborar el desarrollo y cumplimiento del requerimiento.
- **Fecha de revisión**, última fecha de revisión realizada.
- **Resumen**, especificación de la función o funciones que cumple el requerimiento.
- **Entradas**, especificación de datos que recibe el requerimiento.
- **Resultados**, especificación del resultado esperado tras ejecución del requerimiento.
- **Curso básico de eventos**, especificación de un conjunto de acciones ordenadas, que de realizarse se cumple la función del requerimiento.
- **Caminos alternativos**, especificación de acciones alternas, que cumplen la función del requerimiento.
- **Caminos de excepción**, especificación de las condiciones de error existentes en el curso básico de eventos y su tratamiento.
- **Pre - condiciones**, conjunto de acciones previas necesarias para ejecución del requerimiento.
- **Post - condiciones**, estado posterior necesario, tras ejecución del requerimiento.

Se obtuvo un total de 16 requerimientos funcionales descritos en los anexos. A continuación se describe el requerimiento RF-9.

Identificador	RF-9
Nombre del requerimiento	Editar formulario de configuración de funciones CRUD - estándar
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	26/06/2015
Revisión	Héctor Mauricio Cabrera
Fecha de revisión	25/08/2015
Resumen	Usuario especifica función CRUD a crear (insert, select, uptade, delete), nombre del procedimiento, campos y restricciones involucrados en la función. Posteriormente el usuario acciona el evento guardar. Hecho esto el sistema valida y almacena temporalmente la función CRUD especificada.
Entradas	Función CRUD a crear, nombre del procedimiento, campos y restricciones.
Resultados	Almacenamiento temporal de función CRUD en estructura de datos (lista).
Curso Básico de eventos	1. Usuario especifica función CRUD a crear, nombre del procedimiento, selecciona campos y restricciones necesarios en la función. 2. Usuario acciona evento guardar. 3. Sistema valida autenticidad de datos especificados y almacena temporalmente función CRUD.
Caminos alternativos	N/A
Caminos de excepción	1. Punto 3. Si usuario omite la especificación de la función CRUD, nombre del procedimiento o campos, el sistema no efectúa el almacenamiento y espera a que el usuario suministre información correcta.
Pre - condiciones	El usuario debe haber iniciado sesión. El usuario debe haber seleccionado una tabla del esquema de base de datos.
Post - condiciones	Almacenamiento temporal en estructura de datos (lista) de funciones CRUD.

Tabla 3.1: Descripción de requerimiento funcional RF-9

3.1.3. Requerimientos no funcionales

En la descripción de los requerimientos no funcionales se tiene en cuenta la siguiente información:

- **Identificador**, asignación que identifica de forma única al requerimiento.
- **Nombre del requerimiento**, asignación que identifica nominalmente al requerimiento.
- **Tipo**, especifica la clase a la que pertenece el requerimiento. Las clases son: Atributo de arquitectura y restricción tecnológica.
- **Indispensable/Deseable**, determina si requerimiento es indispensable o no.
- **Descripción**, especifica las pautas y limites que establece el requerimiento.
- **Criterios de aceptación**, especifica las condiciones que deben cumplirse para que el requerimiento sea aceptado.

Se obtuvo un total de 2 requerimientos no funcionales descritos en los anexos. A continuación se describe el requerimiento RNF-1.

Identificador	RNF-1
Nombre del requerimiento	Lenguaje de programación
Tipo	Restricción Tecnológica
Indispensable/Deseable	Indispensable
Descripción	El desarrollo del software asociado al proyecto debe estar realizado en los siguientes lenguajes de programación y de marcado: Racket, HTML, CSS, JavaScript.
Criterios de aceptación	La Implementación del proyecto debe hacerse sobre estas tecnologías/-lenguajes.

Tabla 3.2: Descripción de requerimiento RNF-1

3.1.4. Descripción general

Conforme al objetivo general, la herramienta CASE basa su generación de código fuente a partir de un esquema de base de datos previamente establecido, lo que nos conduce a disponer de sus elementos que la constituyen (tablas y campos), dando origen al requerimiento "Desplegar elementos del esquema de base de datos".

La aplicación Web prototipo que se genera, está limitada en estructura y funciones, ocasionando la creación de lineamientos de presentación gráfica y funcionalidades, que son provistos e implementados por el usuario en la construcción de su software. Estos lineamientos obedecen a la posibilidad de asignar valores a las propiedades de cada campo para cada tabla y la creación de funciones CRUD en cada tabla. En conformidad a ello surgen los cuatro requerimientos base "Desplegar tabla dinámica de configuración de campos", "Editar tabla dinámica de configuración de campos", "Desplegar tabla dinámica de funciones CRUD" y "Editar formulario de configuración de funciones CRUD".

El ejercicio anterior considera el almacenamiento temporal (en memoria), sobre las estructuras soportadas por el lenguaje Racket de los datos especificados por el usuario, que nos lleva en forma obligatoria y consecuente al almacenamiento permanente en disco, y por ende la creación de los requerimientos "Almacenar lista de configuración de campos" y "Almacenar lista de configuración de funciones CRUD".

Disponer los valores de las propiedades de los campos, no nos dice más que eso, asignación de valores que deben conservarse, sin corroborar la ejecución de ellos, lo que significa una carencia de información que desconcierta al usuario, en relación a sí lo establecido es lo que realmente desea. Este vacío da origen al requerimiento "Visualizar gráficamente configuración de campos" que ejecuta los valores asignados a los campos e informa visualmente al usuario de lo que ha establecido.

El objetivo de los requerimientos "Exportar aplicación" y "Crear archivos estáticos" ha estado presente desde la concepción original del proyecto, sin embargo su mecanismo para lograrlo sólo fue posible establecer una vez que los requerimientos base y de almacenamiento estuvieron hechos, es decir, los requerimientos hasta ahora descritos. Posteriormente el requerimiento "Crear archivos estáticos" se une al requerimiento "Exportar aplicación" a causa de su condición dependiente e inalterable.

Debido a que el enfoque en la implementación del proyecto estuvo centrada en la generación de código fuente, se desatendió la incorporación de asistencia y ayudas que guian al usuario en el proceso de creación de la aplicación Web prototipo, lo que implicó que esta funcionalidad fuera la última en ser analizada y

en consecuencia su desarrollo mínimo. La asistencia para el proceso de generación de la aplicación Web prototipo, existente en la herramienta, se hace válida a través del requerimiento "Generar mensajes de información".

La privacidad de la información personal, soporte único y consistente para cada uno de los proyectos/esquemas seleccionados por los usuarios, son requisitos indispensables que refuerzan la integridad y confiabilidad de la herramienta, lo que hace esencial dar soporte para el registro de nuevos usuarios y conservar consistentemente sus configuraciones establecidas a través de sesiones. Estas funciones se validan a través de los requerimientos "Registrar usuario", "Iniciar sesión" y "Cerrar sesión".

Los requerimientos que aquí se mencionan pueden examinarse con mayor detalle en los anexos y/o en el capítulo de implementación.

Capítulo 4

Diseño y arquitectura

4.1. Diseño y arquitectura

El diseño y arquitectura en la construcción de software, es el proceso en que se abstrae el sistema de software, se establece la forma en como se dispondrán los elementos y como serán relacionados, con el objetivo de cumplir los requerimientos.

En este capítulo se identifica la arquitectura establecida, la división modular, la definición de estructuras de datos, funciones e interacción entre ellas, el modelo relacional involucrado en el diseño de la herramienta y su modelo de navegación.

4.1.1. Arquitectura de mútiples capas

La herramienta CASE es de hecho un sistema, compuesto de tres subsistemas dispuestos ordenadamente, que interactúan y colaboran entre sí, para contribuir en el cumplimiento de sus objetivos. Los tres subsistemas que definen a la herramienta son: *recolección de datos*, *generación de datos* y *almacenamiento de datos*.

Recolección de datos

Encargado de recolectar los datos especificados por el usuario en el proceso de generación de la aplicación prototipo. Se comunica con la capa *Almacenamiento de datos* para obtener la información del esquema implicado en la generación y con la capa *Generación de datos* para suministrarle los datos otorgados por el usuario.

Los datos recolectados son: Configuración de campos, configuración de funciones CRUD y configuración de funciones CRUD personalizadas. Sus componentes son: codegen. (Ver en sección 5.1.2 la descripción de cada componente).

Generación de datos

Encargado de disponer los datos recolectados en una estructura de datos modificable, almacenarlos de forma permanente en archivos y generar a través de ellos la aplicación prototipo. Sus componentes son:

generador, tablas, campos, crud, crud-funcs, crud-render, crud-format, view-funcs y conf. (Ver en sección 5.1.2 la descripción de cada componente).

Almacenamiento de datos

Encargado de acceder a la base de datos para conservar de forma permanente los datos del usuario y del esquema de base de datos provisto para la generación. Sus componentes son: conn. (Ver en sección 5.1.2 la descripción de cada componente).

Con base en la notación UML 2.0 para los *diagramas de paquetes*, se presenta a continuación el *diagrama de arquitectura multicapa* con sus componentes y relaciones.

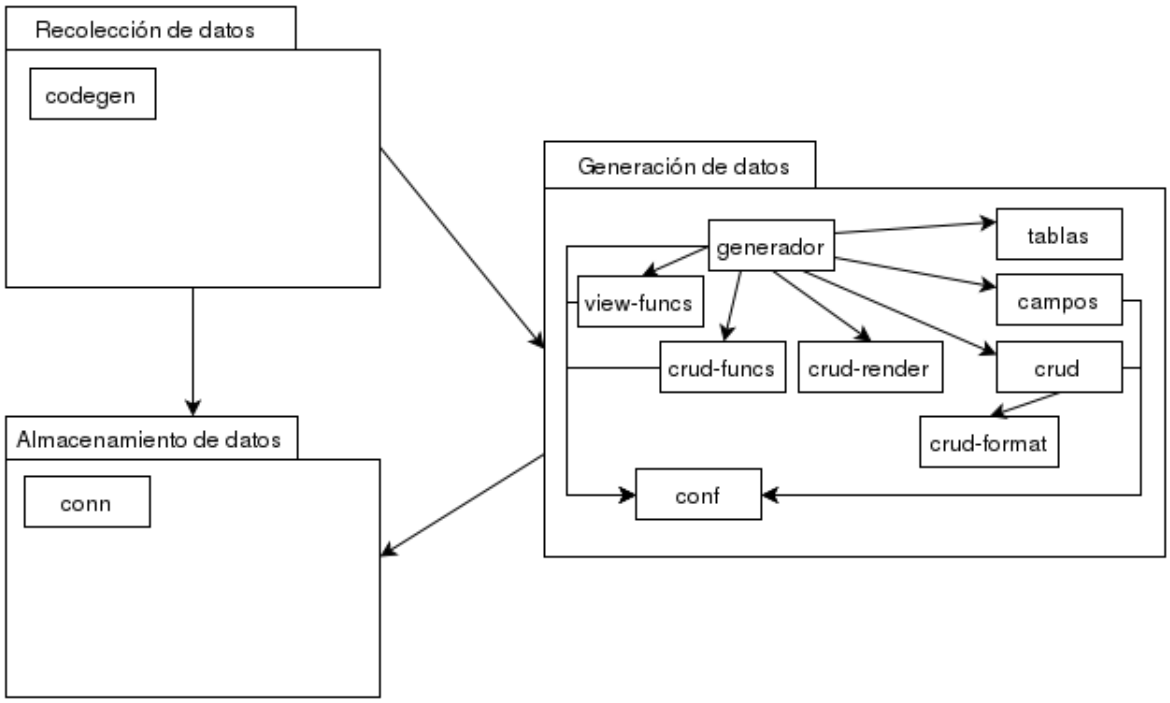


Figura 4.1: Arquitectura multicapa

4.1.2. Módulos

Conforme a los objetivos específicos, y debido a la facilidad que otorga para el reconocimiento de acciones específicas, se decide crear módulos capaces de cumplir con un grupo de tareas relacionadas. Los módulos creados para el desarrollo de la herramienta son:

Módulos de gestión de configuración de campos

Encargados de capturar, procesar, almacenar y desplegar las configuraciones de las propiedades de los campos para cada tabla. Los módulos son: codegen, tablas y campos. (Ver descripción de cada módulo en la sección siguiente).

Módulos de gestión de funciones CRUD

Encargados de capturar, procesar, almacenar y desplegar las configuraciones de las funciones CRUD para cada tabla. Los módulos son: codegen y crud. (Ver descripción de cada módulo en la sección siguiente).

Módulos de gestión de generación de código fuente

Encargados de recuperar información de configuración de campos y configuración de funciones CRUD de la totalidad de las tablas y generar aplicacion Web prototipo. Los módulos son: generador, crud-format, crud-funcs, crud-render, view-funcs y conf. (Ver descripción de cada módulo en la sección siguiente).

Módulo de gestión de persistencia

Encargado de acceder a la base de datos, consultar y registrar datos del usuario y elementos del esquema de base de datos. El módulo es conn. (Ver descripción de cada módulo en la sección siguiente).

Se describe a continuación la reponsabilidad asignada a cada módulo o componente.

- **codegen:** Es la interfaz gráfica que comunica al usuario con la herramienta CASE, encargado de desplegar la información perteneciente al esquema de base de datos (tablas y campos), cargar tabla dinámica de configuración de campos y formularios de configuración de funciones CRUD. Finalmente es responsable de direccionar toda petición del usuario, a los módulos adecuados para su correcto tratamiento.
- **tablas:** Encargado de consultar las tablas pertenecientes al esquema y almacenarlas en estructura de datos para su posterior uso por parte de la herramienta. Controla además la estructura de datos que lista las tablas que deben ser generadas en la aplicación prototipo.
- **campos:** Encargado de consultar los campos de la totalidad de las tablas del esquema, bien sea desde archivo o desde la base de datos, agregar propiedades y valores por defecto para crear estructura de configuración de campos. Captura, prepara y almacena información de configuración de campos especificada por el usuario en archivo. Finalmente dispone de las funciones que módulo codegen usa para el despliegue de la tabla dinámica de configuración de campos.
- **crud:** Encargado de capturar, cargar y/o modificar en estructura de datos las configuraciones de las funciones CRUD especificadas por el usuario. Finalmente almacena en archivo las configuraciones de funciones CRUD hechas.
- **crud-format:** Encargado de formatear adecuadamente las configuraciones de funciones CRUD según su tipo, para su posterior uso en la generación de la aplicación prototipo. (Formación de intrucciones SQL en cadena de texto con campos y restricciones especificadas).
- **crud-funcs:** Encargado de construir el archivo de persistencia de la aplicación prototipo. (Especificación de las funciones CRUD con información previamente formateada).
- **crud-render:** Encargado de especificar los formularios Web, funciones de captura de datos y de llamado a funciones CRUD de la aplicación prototipo.

- **view-funcs:** Encargado de construir archivo con funciones de presentación de datos de la aplicación prototipo.
- **generador:** Encargado de direccionar la creación de la aplicación prototipo. Permite la generación de las páginas Web por cada tabla activa, formularios y funciones CRUD asociadas, funciones de presentación y establece las relaciones de dependencia entre los archivos generados. En su proceso utiliza las configuraciones de los campos y funciones CRUD realizadas.
- **conn:** Encargado acceder a la base de datos, consultar y registrar datos del usuario y elementos del esquema de base de datos.
- **conf:** Encargado de definir valores constantes utilizados por la herramienta.

4.1.3. Estructuras de datos

Para permitir que la información suministrada por el usuario a través de la herramienta, permanezca consistente y disponible, es necesario conservarla en una estructura de datos permitida por el lenguaje de programación. A continuación se lista las características de dicha estructura:

- **Denominación**, nombre que caracteriza a la estructura de datos.
- **Denominación en el lenguaje**, nombre que caracteriza a la estructura en el lenguaje de programación.
- **Descripción**, especificación verbal de la estructura.
- **Nombre en implementación**, asignación que identifica a la estructura en la implementación de la herramienta.
- **Nombre descriptivo**, asignación que describe su función en la herramienta.
- **Configuración**, descripción interna de la estructura.
- **Ejemplo**, ejemplificación del uso de la estructura.
- **Uso**, descripción de la función que desempeña en la herramienta.

Acorde a las características mencionadas, se especifica a continuación las estructuras de datos implementadas en la herramienta:

Denominación	Lista de listas
Denominación en lenguaje	(listof list?)
Descripción	Lista que contiene un conjunto de listas, cuyos elementos pueden ser de cualquier tipo.
Nombre en implementación	vcamposg
Nombre descriptivo	Lista de configuración de campos

Configuración	(list (list v) ...) v : string - nombre del esquema string - nombre de tabla string - nombre de etiqueta de campo string - nombre de campo string - tipo de elemento string - tipo de restricción string - tamaño lógico string - tamaño físico string - nulable string - mostrar (list (list w) ...) - lista de opciones w : string - valor a almacenar string - valor a mostrar
Ejemplo	(("empven" "ventas" "Total venta" "totalventa" "text" "numerico" "10" "10" "NO" "t" ()) ("empven" "ventas" "Fecha de venta" "fecventa" "select_date" "alfanumerico" "10" "10" "NO" "t" ()) ("empven" "ventas" "Numero documento" "numdoc" "text" "numerico" "10" "10" "NO" "t" ()) ("empven" "ventas" "idventa" "id" "text" "alfanumerico" "10" "10" "NO" "f" ()))
Descripción de ejemplo	Establece la configuración de los campos para tabla *ventas* del esquema *empven*. El campo totalventa se define como tipo texto, restringido a números, tamaño lógico 10, con etiqueta *Total venta* y es visible. El campo fecventa se define como tipo fecha, restricción alfanumérico, tamaño lógico 10, con etiqueta *Fecha de venta* y es visible. Los demás campos se definen bajo la misma estructura, cada uno con su respectivo valor.
Uso	Implementada para conservar las configuraciones de los campos de la totalidad de las tablas del esquema de base de datos.

Tabla 4.1: Descripción de estructura - Lista de configuración de campos

Denominación	Lista de listas
Denominación en lenguaje	(listof list?)
Descripción	Lista que contiene un conjunto de listas, cuyos elementos pueden ser de cualquier tipo.
Nombre en implementación	vcrudg
Nombre descriptivo	Lista de configuración de funciones CRUD

Configuración	(list (list v) ...) v : string - nombre del esquema string - nombre de tabla string w - función CRUD La cadena de texto w presenta el siguiente formato: nombre-del-esquema & nombre-de-tabla & nombre-de-función & tipo-de-función-SQL & campo1,campo2,...,campoN & restricciones
Ejemplo	(("empven" "ventas" "AND & empven & ventas & select-all-venta & SELECT & totalventa,fecventa,numdoc") ("empven" "ventas" "AND & empven & ventas & insert-venta & INSERT & totalventa,fecventa,numdoc"))
Descripción de ejemplo	Establece la configuración de funciones CRUD para tabla *ventas* del esquema *empven*. Se establece instrucción SQL de tipo SELECT, con nombre de procedimiento *select-all-venta* y campos totalventa, fecventa y numdoc. Se establece instrucción SQL de tipo INSERT, con nombre de procedimiento *insert-venta* y campos totalventa, fecventa y numdoc.
Uso	Implementada para conservar las configuraciones de las funciones CRUD de la totalidad de las tablas del esquema de base de datos.

Tabla 4.2: Descripción de estructura - Lista de configuración de funciones CRUD

Denominación	Lista de listas
Denominación en lenguaje	(listof list?)
Descripción	Lista que contiene un conjunto de listas, cuyos elementos pueden ser de cualquier tipo.
Nombre en implementación	vcrudg-per
Nombre descriptivo	Lista de configuración de funciones CRUD personalizadas
Configuración	(list (list v) ...) v : string - nombre del esquema string - nombre de tabla string w - función CRUD personalizada La cadena de texto w presenta el siguiente formato: nombre-del-esquema & nombre-de-tabla & nombre-de-función & tipo-de-función-SQL & instrucción-SQL & campo1,campo2,...,campoN [&] restricciones

Ejemplo	((“empven” “ventas” “empven & ventas & update-venta & UPDATE & update empven.ventas set totalventa=$1 where numdoc=$2 and id=$3 & totalventa,numdoc,id”))
Descripción de ejemplo	Establece la configuración de funciones CRUD personalizadas para tabla *ventas* del esquema *empven*. Se establece instrucción SQL de tipo UPDATE, con nombre de procedimiento *update-venta*, campo a actualizar totalventa y campos restricción numdoc e id.
Uso	Implementada para conservar las configuraciones de las funciones CRUD personalizadas de la totalidad de las tablas del esquema de base de datos.

Tabla 4.3: Descripción de estructura - Lista de configuración de funciones CRUD personalizadas

4.1.4. Archivos de persistencia

Para permitir que la información especificada por el usuario y soportada en las estructuras de datos, se conserve de forma permanente una vez que el usuario abandone la herramienta, es necesario utilizar un medio de almacenamiento. El medio utilizado es el archivo plano. La prevalencia de este medio en lugar de la base de datos, es debido principalmente a su fácil integración con la herramienta. A continuación se describe algunas propiedades de dicho archivo:

- **Nombre**, asignación que identifica al archivo.
- **Tipo**, asignación que especifica el tipo de archivo.
- **Descripción**, especifica la función que cumple el archivo.
- **Contenido**, especifica la estructura de datos que almacena el archivo.

Acorde a la descripción anterior, se muestra a continuación los archivos de persistencia que fueron implementados en la herramienta.

Nombre	nombre-esquema.data
Tipo	Archivo de texto plano
Descripción	Archivo con las configuraciones de los campos de la totalidad de las tablas.
Contenido	vcamposg

Tabla 4.4: Descripción de archivo de persistencia - Para lista de configuración de campos

Nombre	nombre-esquema.crud
Tipo	Archivo de texto plano
Descripción	Archivo con las configuraciones de las funciones CRUD de la totalidad de las tablas.
Contenido	vcrudg

Tabla 4.5: Descripción de archivo de persistencia - Para lista de configuración de funciones CRUD

Nombre	nombre-esquema.crudp
Tipo	Archivo de texto plano
Descripción	Archivo con las configuraciones de las funciones CRUD personalizadas de la totalidad de las tablas.
Contenido	vcrudg-per

Tabla 4.6: Descripción de archivo de persistencia - Para lista de configuración de funciones CRUD personalizadas

4.1.5. Funciones e interacción

Las funciones son el principal medio sobre el cual se desarrolla la herramienta, a través de cada una se asigna una tarea en específico y junto con la composición adecuada se construye y cubre cada uno de los requerimientos.

Se describe a continuación algunas de las funciones y su interacción necesaria para el cumplimiento de un requerimiento. En la descripción se tiene en cuenta la siguiente información:

- **Módulo:** Especifica a que módulo pertenece.
- **Contrato:** Especifica nombre de la función, cuantos parámetros recibe, de que tipo y que retorna.
- **Definición en implementación:** Especifica como se ha definido en la implementación de la herramienta.
- **Descripción:** Especifica de que es responsable la función.

Las funciones que se describen a continuación pertenecen al requerimiento RF-6 Editar tabla dinámica de configuración de campos.

codegen
modf-table :: request → expr?
(modf-table request)
Captura nombre de tabla, configuración de campos y modifica estructura de datos vcamposg para almacenar la nueva información. Retorna expresión.

campos
prepare-new-lst :: string string list? → (listof list?)
(prepare-new-lst schema table str)
Toma expresión en cadena de caracteres proveniente de la tabla de configuración de campos editada por el usuario y la transforma en lista de listas. Retorna lista de listas.

campos
fix-fields :: list? → (listof list?)
(fix-fields lst)
Construye lista de listas de las propiedades de los campos de una tabla del esquema. Usa para ello la información proveniente de la tabla de configuración de campos editada por el usuario.

campos
modf-lst :: string string (listof list?) → void?
(modf-lst schema table lst)
Modifica las propiedades de los campos de tabla especificada sobre estructura vcamposg. Retorna void?.

campos
clean-lst :: string (listof list?) → (listof list?)
(clena-lst table lst)
Elimina las propiedades de los campos de tabla especificada sobre estructura vcamposg. Retorna lista de listas.

campos
add-elemts :: (listof list?) → (listof list?)
(add-elemts lst)
Agrega las propiedades de los campos de tabla especificada sobre estructura vcamposg. Retorna lista de listas.

codegen
codegen-index :: string string string request → expr?
(codegen-index id schema ntable request)
Despliega página principal de la herramienta CASE. Lista tablas del esquema, crea tabla de configuración de campos, tablas de funciones CRUD y formularios de configuración de funciones CRUD. Retorna expresión.

Con base en la notación UML 2.0 para los *diagramas de secuencia*, se presenta a continuación un diagrama similar que demuestra la relación existente entre las funciones y su procedimiento. La especificación de 213 funciones y 9 Diagramas de secuencia en total, se encuentra en los anexos.

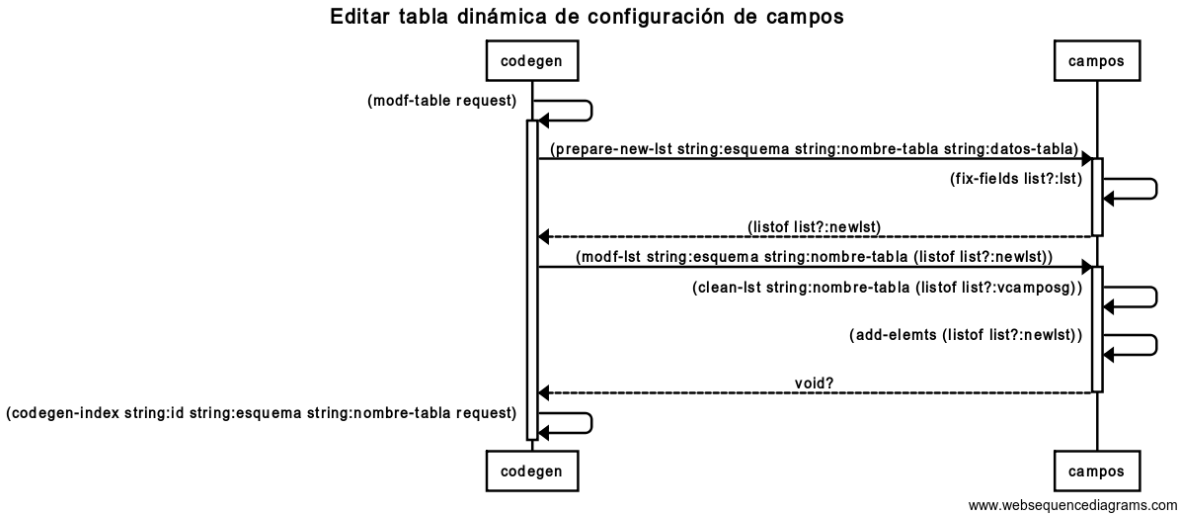


Figura 4.2: Diagrama de secuencia - Editar tabla dinámica de configuración de campos

4.2. Modelo relacional de base de datos

La herramienta CASE cuenta con una base de datos denominada *codegen*, sobre la que se hacen consultas para resolver requerimientos de función. Para llegar al modelo relacional se parte del siguiente modelo E-R.

- **Entidad usuario**: Especifica los datos personales de cada usuario registrado. Sus atributos son (correo-electrónico, nombre, apellidos, fecha-nacimiento, sexo, contraseña, conf-contraseña).
- **Entidad esquema**: Especifica los nombres de los esquemas registrados. Su atributo es nombre-esquema.
- **Relación tiene**: Relaciona cada usuario con un esquema sobre el que desea trabajar.

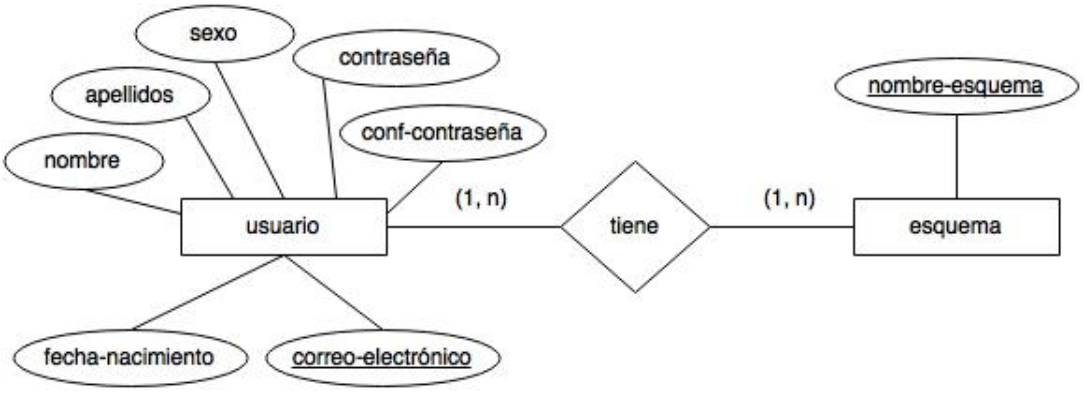


Figura 4.3: Diagrama Entidad-Relación para entidades usuario y esquema

El modelo relacional que se deriva del modelo E-R especificado, se muestra a continuación.

USUARIO(correo-eletrónico, nombre, apellidos, fecha-nacimiento, sexo, contraseña, conf-contraseña)
ESQUEMA(nombre-esquema)
TIENE(correo-electrónico, nombre-esquema)
Clave externa: correo-electrónico - Hace referencia a USUARIO.
Clave externa: nombre-esquema - Hace referencia a ESQUEMA.
Clave compuesta: correo-electrónico, nombre-esquema

4.3. Modelo de navegación

Para comprender la estructura de navegación de la herramienta se hace uso de una representación gráfica. Es necesario tener en cuenta que la navegación en la estructura, depende del esquema de base de datos seleccionado por el usuario en el momento de acceder a la herramienta. Es decir, aunque el modelo

de navegación y los mecanismos de acción persisten, las tablas a seleccionar dependerán del esquema elegido. El diagrama que se presenta a continuación describe la estructura de navegación y forma de proceder de la herramienta independiente del esquema seleccionado.

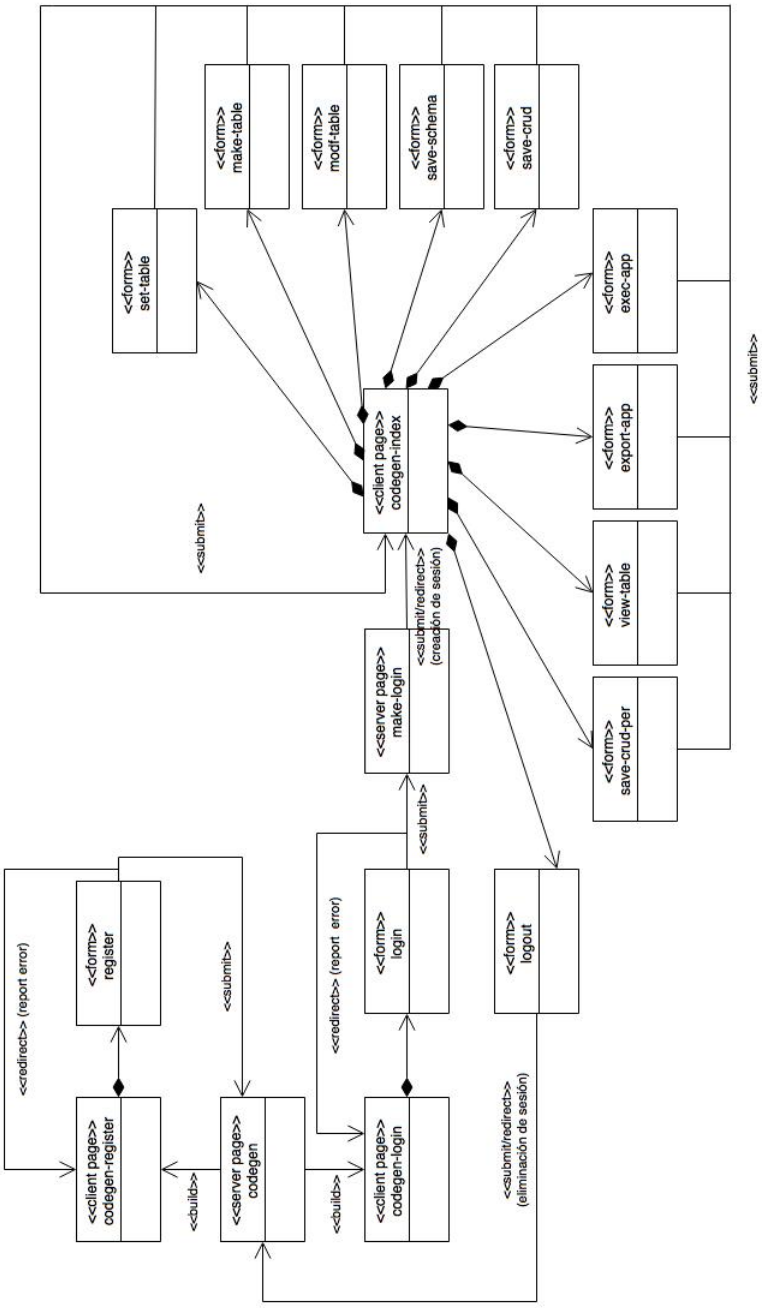


Figura 4.4: Modelo de navegación

Capítulo 5

Implementación

5.1. Implementación

Una vez definido las capacidades del software, límites y restricciones de su funcionamiento y estructura de sus elementos, se procede con la implementación del código fuente que sea capaz de interpretar y satisfacer los requerimientos impuestos.

En este capítulo se van a explicar los aspectos de implementación más relevantes de la herramienta. En la explicación se añadirá diagramas y/o bloques de código fuente para demostrar explícitamente el trabajo realizado.

5.1.1. Modelo relacional para esquema empven

Antes de especificar los detalles de implementación, es necesario disponer de un esquema de base de datos sobre el cual la herramienta base sus funciones. Es decir, utilice sus elementos (tablas y campos) como fuente de información para el proceso de generación de código. El modelo E-R y modelo relacional que se describe a continuación conforman el diseño previo para el esquema *empven*, que será implementado en la base de datos *codegen*.

Las entidades, atributos y relaciones de este modelo se describen a continuación.

- **Entidad dpto**: Dispone información de los departamentos. Sus atributos son (id-dpto, descripción).
- **Entidad ciudad**: Dispone información de las ciudades. Sus atributos son (id-ciudad, descripción).
- **Entidad empleado**: Dispone información personal de los empleados. Sus atributos son (número-documento, nombre, apellidos, tipo-documento, fecha-nacimiento, sexo, correo-electrónico, dirección).
- **Entidad ventas**: Dispone información de las ventas de cada empleado. Sus atributos son (id-venta, fecha-venta, total-venta).
- **Relación pertenece**: Relaciona entidades dpto y ciudad.
- **Relación vive**: Relaciona entidades empleado y ciudad.

- **Relación tiene**: Relaciona entidades empleado y ventas.

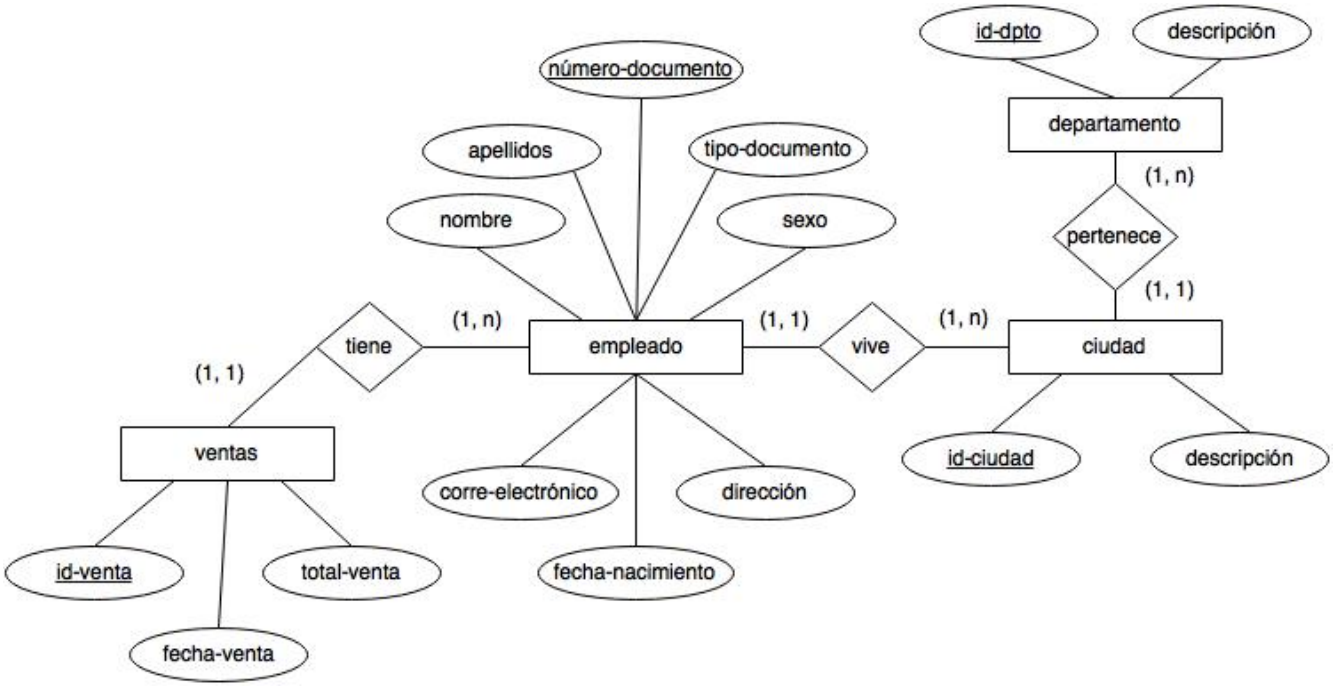


Figura 5.1: Diagrama Entidad-Relación para entidades dpto, cuidad, empleado y ventas

El modelo relacional que se deriva del modelo E-R descrito, se especifica a continuación.

DPTO(id-dpto, descripción)
CIUDAD(id-ciudad, id-dpto, descripción)
Clave externa: id-dpto - Hace referencia a DPTO.

EMPLEADO(número-documento, nombre, apellidos, tipo-documento, fecha-nacimiento, sexo, correo-eletrónico, dirección, id-dpto, id-ciudad)
Clave externa: id-dpto - Hace referencia a DPTO.
Clave externa: id-ciudad - Hace referencia a CIUDAD.

VENTAS(id-venta, número-documento, fecha-venta, total-venta)
Clave externa: número-documento - Hace referencia a EMPLEADO.

5.1.2. Configuración del Servidor Web

Una vez creada la base de datos de la herramienta, establecido el esquema sobre el cual trabajar (véase manual de instalación), se procede a configurar el servidor Web sobre el cual se despliega la herramienta. La configuración se logra a través de la importación de módulos especializados del lenguaje Racket y la especificación de propiedades únicas del servidor. Esta configuración se muestra en el siguiente bloque de código.

Código Fuente 5.1: Importación de módulos - Módulo codegen

```
#lang racket

; Módulos especializados para configuración del Servidor Web

(require web-server/servlet
         web-server/servlet-env
         web-server/dispatch)
(require (planet neil/html-writing:2:0))

(provide/contract (start (request? . -> . response?)))

(require 2htdp/batch-io)
```

Código Fuente 5.2: Configuración del Servidor Web - Módulo codegen

```
; Configuración del Servidor Web

; Ruta donde reside configuración del servidor y aplicación web.
(define server-path-str (path->string (current-directory)))
(define server-path (build-path server-path-str))

(serve/servlet start
    #:launch-browser? #f
    #:quit? #f
    #:listen-ip #f
    #:port 8000
    #:server-root-path server-path
    #:extra-files-paths (list (build-path server-path-str "htdocs"))
    #:servlet-path "/codegen"
    #:servlet-regexp #rx""
    #:file-not-found-responder codegen-login-tmp)
```

Las propiedades que se especifican en la configuración del Servidor Web tienen el siguiente significado:

- **#:launch-browser?** permite determinar si se lanza un navegador web una vez se ejecute el servidor web.
- **#:quit?** determina si se debe finalizar el servidor web con URL /quit.
- **#:listen-ip** determina si se debe aceptar conexiones desde cualquier dirección IP o alguna en específico.
- **#:port** determina el puerto sobre el cual se ejecuta la aplicación web.
- **#:server-root-path** determina la ruta en el sistema de archivos donde reside la configuración del servidor web.

- **#:extra-files-paths** determina la ruta en el sistema de archivos donde residen los archivos estáticos de la aplicación web.
- **#:servlet-path** y **#:servlet-regexp** determinan la forma de URL para el acceso a la aplicación web.
- **#:file-not-found-responder** determina la página web a cargar, en caso de que especificación de URL no corresponde a ninguna regla de despacho.

5.1.3. Conexión a base de datos

Una vez se logra el despliegue del servidor web se continúa con el establecimiento de la conexión a la base de datos. El propósito además de la conexión, es conservar el estado "conectado" mientras el usuario se encuentre activo en la herramienta. Esto se logra a través de las funciones *start*, *inicializar-conn* e *inicializado*.

Código Fuente 5.3: Iniciar conexión a base de datos - Módulo codegen

```
(define (start request)
    (if (not (inicializado))
        (let ((inic-app (inicializar-conn)))
            (if inic-app
                (codegen-dispatch request)
                #f))
        (codegen-dispatch request)))
```

El procedimiento *start* se especifica para dar tratamiento a la primera petición de la aplicación web y su definición es de carácter obligatorio. Para hacer el despliegue de la herramienta CASE, es necesario estar conectado a la base de datos, el contenido del procedimiento *start* ayuda a realizar y verificar dicha conexión. La manera en que se comporta este procedimiento es el siguiente: Si no se ha realizado la conexión, se obtiene #f de la llamada a la función *(inicializado)*, se inicia la conexión a la base de datos a través de función *(inicializar-conn)*. Si la conexión se logra, se inicia el despliegue de página web solicitada a través de *(codegen-dispatch request)*, de lo contrario se retorna #f. Si conexión ya ha sido realizada, se inicia el despliegue de página web solicitada.

Código Fuente 5.4: Conexión a base de datos - Módulo conn

```
#lang racket/base
(require racket/list
                racket/local
                racket/file
                db)

; Variable global para conocer estado de la conexión
; con la base de datos.
(define db #f)
```

A través de *require* se importan los módulos *list*, *local*, *file* y *db* que proveen funciones para operar con listas, definiciones locales, archivos y bases de datos respectivamente, no provistos por la definición *racket/base*.

Código Fuente 5.5: Conexión a base de datos - Módulo conn

```
; inicializar-conn :: nothing -> boolean
; Establece conexión con base de datos, de ser así
; devuelve #t, de lo contrario #f.
(define (inicializar-conn)
        (set! db (postgresql-connect
                        #:server "localhost"
                        #:port 5432
                        #:database "codegen"
                        #:user "postgres"
                        #:password "56648865"))
        (if (connected? db)
                #t
                #f))

; inicializado :: nothing -> boolean
; Verifica si existe conexión con base de datos establecida,
; si es así devuelve #t, de lo contrario #f.
(define (inicializado)
        (if (and (not (boolean? db)) (connected? db))
                #t
                #f))
```

Para la conexión con la base de datos es necesario especificar las siguientes propiedades:

- **#:server** determina ubicación del servidor, por defecto es localhost.
- **#:port** determina puerto sobre el cual se ejecuta la base de datos.
- **#:database** determina el nombre de la base de datos a la que se desea conectar.
- **#:user** determina usuario válido de la base de datos.
- **#:password** determina contraseña del usuario de la base de datos.

5.1.4. Elementos de los esquemas de base de datos

La principal fuente de información para el propósito de la herramienta son los elementos (tablas y campos) de los esquemas de base de datos. Estos elementos se consultan en la base de datos a través de sentencias SQL por intermedio de funciones del lenguaje de programación Racket. Las sentencias SQL en este caso son intrínsecas del gestor de base de datos, por lo que la especificación de las sentencias podría variar de un gestor a otro. Las sentencias SQL que se emplearon en base al gestor utilizado son:

Código Fuente 5.6: Consulta de tablas de un esquema - Módulo conn

```
SELECT tablename FROM pg_tables
    WHERE schemaname = 'nombre_de_esquema'
```

Código Fuente 5.7: Consulta de campos de una tabla - Módulo conn

```
SELECT table_schema, table_name, column_name, data_type, character_maximum_length, is_nullable,
    ordinal_position
    FROM information_schema.columns
    WHERE table_schema = 'nombre_de_esquema' and
        table_name = 'nombre_de_tabla'
```

En ambas sentencias SQL los indicadores *nombre_de_esquema* y *nombre_de_tabla* deben ser reemplazados por el nombre de esquema y tablas previamente establecidos en la base de datos. En este caso los valores posibles son *empven* como nombre de esquema y *empleado, dpto, ciudad, ventas* como nombre de tabla.

5.1.5. Creación de archivos de estáticos

Para la disposición gráfica de la herramienta, tratamiento de los datos de entrada e interacción con el usuario, es necesario disponer de directrices que formen y guien dicho proceso. Las directrices se encuentran especificadas como funciones o reglas en archivos estáticos acordes a su lenguaje, Javascript y CSS respectivamente. A continuación se describe la función de cada archivo estático en la herramienta.

- **codegen.js** - Encargado de accionar algunas de las funciones de la herramienta (make_table, save_schema, export_app, view_table), desplegar información de sesión y mensajes de información.
- **codegen-start.js** - Encargado de restringir los campos de los formularios de registro e inicio de sesión a los valores permitidos y de accionar su envío para su procesamiento.
- **codegen-tablaconf.js** - Encargado de construir gráficamente la tabla dinámica de configuración de campos, preparar y conservar las configuraciones especificadas por el usuario en el momento de edición.
- **codegen-settable.js** - Encargado de soportar gráficamente el estado de las tablas del esquema de base de datos que están o no activas. Acciona función para almacenar estado de las tablas.
- **codegen-crudform.js** - Encargado de construir gráficamente tablas de funciones CRUD, mostrar formularios de configuración de funciones CRUD, preparar y conservar las funciones CRUD especificadas por el usuario en el momento de edición, y accionar función para almacenar de forma permanente las funciones CRUD de las tablas. Las funciones que aquí se describen aplican tanto para funciones CRUD estándar y personalizadas.
- **codegen-crudpers.js** - Encargado de controlar los eventos necesarios para la formación de las funciones CRUD personalizadas. (autocompletado, formación de lista de campos y/o restricciones, captura de eventos del teclado, etc).
- **app.js** - Encargado de reposicionar gráficamente los campos de las páginas de la aplicación web prototipo y restringir los valores de cada campo al valor permitido.
- **codegen.css** - Encargado de establecer las reglas de distribución y tamaño de los elementos gráficos de la herramienta.
- **app.css** - Encargado de establecer las reglas de distribución y tamaño de los elementos gráficos de la aplicación web prototipo.

5.1.6. Creación de lista de configuración de campos

Una vez obtenidas las propiedades base de los campos de cada tabla del esquema, se construye una estructura de datos capaz de almacenar la información de forma temporal, mientras el usuario se encuentra activo en la herramienta. En la creación se añaden nuevas propiedades con valores por defecto necesarios para la herramienta. La creación de esta estructura se logra a través de la función *lst-conf*. La elección de listas como representación de las estructuras de datos obedece a la reducción de código necesario para acceder, modificar y almacenar los datos en si mismas y en los archivos, sacrificando la libre expresividad que otorgan las estructuras.

La estructura de datos que se construye en este procedimiento posteriormente se asigna a variable global vcamposg. Descrita en la Tabla 4.1 del Capítulo 4 Diseño y Arquitectura.

Código Fuente 5.8: Creación de lista de configuración de campos - Módulo campos

```
; lst-conf :: (listof list?) -> (listof list?)
; Se toma lista de listas de las propiedades de los campos de cada
; tabla generada inicialmente desde la base de datos y se
; complementan con otras propiedades. Retorna lista de listas.
(define (lst-conf lsta)
  (if (empty? lsta)
    '()
    (let* ((lst (first lsta))
           (schema (first lst))
           (ntabla (second lst))
           (ncampo (third lst))
           (label ncampo)
           (tipoelemt (get-tipo-elemt (fourth lst)))
           (restr "alfanumerico")
           (taml (get-tam-elemt tipoelemt (fifth lst)))
           (tamf taml)
           (nulable (sixth lst))
           (show "t"))
      (cons
        (list schema ntabla label ncampo tipoelemt restr taml tamf nulable show '())
          (lst-conf (rest lsta))))))
```

5.1.7. Creación de lista de funciones CRUD

Para la creación de la estructura de datos de funciones CRUD el procedimiento cambia debido a que su información proviene directamente del usuario a través del formulario de configuración de campos, comprometiendo funciones adicionales para la preparación de los datos. Las funciones que permiten la creación de esta estructura son: *prepare-data-crud*, *clean-crud* y *add-crud*.

La estructura de datos que se construye a través de estos procedimientos posteriormente se asigna a variable global vcrudg. Descrita en la Tabla 4.2 del Capítulo 4 Diseño y Arquitectura.

Código Fuente 5.9: Creación de lista de funciones CRUD

```
; prepare-data-crud :: string string string -> (listof list?)
; Toma expresión en cadena de caracteres proveniente del
; formulario de configuración de funciones CRUD creada por
; el usuario y la transforma en lista de listas.
; Retorna lista de listas.
(define (prepare-data-crud schema ntable datacrud)
    (define precrud (regexp-split #rx"%" datacrud))
    (define lstcrud (map (lambda (elemt)
                           (list schema ntable elemt))
                         precrud))
    lstcrud)

; clean-crud :: string (listof list?) -> (listof list?)
; Elimina configuraciones de funciones CRUD de una tabla en
; específico. Retorna lista de listas.
(define (clean-crud ntable lst)
    (if (empty? lst)
        '()
        (let* ((lsta (first lst))
               (table (second lsta)))
            (if (string=? table ntable)
                (clean-crud ntable (rest lst))
                (cons lsta (clean-crud ntable (rest lst)))))))

; add-crud :: (listof list) -> (listof list)
; Adiciona lista de configuraciones de funciones CRUD de una
; tabla en específico. Retorna lista de listas.
(define (add-crud lst)
    (if (empty? lst)
        vcrudg
        (cons (first lst) (add-crud (rest lst)))))
```

5.1.8. Almacenamiento de lista de configuración de campos

Para lograr el almacenamiento permanente de las configuraciones de los campos hecha por el usuario y que reside hasta ahora en una estructura de datos, es necesario almacenarla en un archivo. El almacenamiento se logra a través de la función *save-schema-file*.

Código Fuente 5.10: Almacenamiento de lista de configuración de campos

```
; save-schema-file :: string -> void?
; Almacena en archivo schema.data los valores contenidos en
; vcamposg. Retorna void?.
(define (save-schema-file schema)
  (if (not (empty? vcamposg))
    (let* ((file (string-append schema-dir schema ".data"))
           (out  (open-output-file file
                 #:mode 'binary
                 #:exists 'replace)))
        (write vcamposg out)
        (close-output-port out))
        (display "\nsave-schema-file: uncharged scheme. file was not created.\n")))
```

El archivo que aquí se genera es el que se encuentra descrito en la Tabla 4.4 del Capítulo 4 Diseño y Arquitectura.

5.1.9. Almacenamiento de lista de funciones CRUD

Con el mismo propósito se almacena la estructura de datos de las funciones CRUD en archivo. Esta función junto con la mencionada anteriormente asegura que tanto las configuraciones de campos y funciones CRUD permanezcan una vez el usuario abandone la herramienta. El almacenamiento de las funciones CRUD se logra a través de la función *save-crud-file*.

Código Fuente 5.11: Almacenamiento de lista de configuración de funciones CRUD

```
; save-crud-file :: string -> void?
; Almacena en archivo schema.crud las configuraciones de funciones
; CRUD de la totalidad de las tablas del esquema. Retorna void?.
(define (save-crud-file schema)
    (if (not (empty? vcrudg))
        (let* ((file (string-append crud-dir schema ".crud"))
               (out (open-output-file file
                         #:mode 'binary
                         #:exists 'replace)))
            (write vcrudg out)
            (close-output-port out))
        (display "\nsave-crud-file: uncharged crud. file was not created.\n")))
```

El archivo que aquí se genera es el que se encuentra descrito en la Tabla 4.5 del Capítulo 4 Diseño y Arquitectura.

En ambos casos para la creación del archivo se debe especificar además del nombre del archivo las siguientes propiedades: #:mode determina el tipo de archivo a crear. #:exists determina acción a realizar si archivo existe.

5.1.10. Generación de páginas de aplicación prototipo

Llegado a este punto se inicia el proceso de generación de código fuente de la aplicación web protopipo. Una tarea de extrema importancia es la generación de las páginas web por cada tabla del esquema activa, incluyendo la configuración de sus campos, formularios y llamadas a funciones CRUD realizadas por el usuario.

Para este proceso se hizo uso de una macro (Ver sección 4.1.1 Generación de código fuente) que extiende su definición para la inclusión, entrega y ejecución de nueva sintaxis, que comprende las funciones para crear los formularios, funciones de captura de datos y de llamado a funciones CRUD especificadas por el usuario.

La macro se ha denominado *make-page* y recibe como parámetros el nombre del esquema, nombre de la página y nombre de la tabla. En su ejecución la macro se expande en las funciones *render-page* y *func-sql-render* para crear las páginas junto con los formularios y las funciones de captura y llamado respectivamente.

Algo importante a tener en cuenta es que en la generación de código fuente se utilizan expresiones de Racket directamente, en vez de concatenar e imprimir cadenas de texto que van formando el código.

El proceso descrito anteriormente se logra a través de las funciones *make-page*, *render-page* y *func-sql-render*.

```
; make-page :: syntax -> syntax
; Macro que extiende su definición para generar función
; encargada de especificar y crear página web de una tabla en
; específico, usando en el proceso la configuración hecha a los
; campos y funciones CRUD. Retorna sintaxis.
(define-syntax (make-page stx)
  (syntax-case stx ()
    [(_ sch npg ntb)
     (if (and (identifier? #'npg) (identifier? #'ntb))
         #'`(define (,(string->symbol npg) args request)
              (local [(define (response-generator embed/url)
                 (response/xexpr
                  ,(list 'quasiquote (render-page sch npg ntb))))
                     ,@(func-sql-render npg ntb)
                     ]
                (send/suspend/dispatch response-generator)))
         (raise-syntax-error #f "No es un identificador." stx #'ntb))
     ]))
```

A través de la sintaxis *make-page* que toma como argumentos el nombre del esquema *sch*, nombre de la página *npg* y nombre de la tabla *ntb*, se retorna una nueva sintaxis extendida que define una nueva página web a través de *render-page* y llamadas a funciones de persistencia a través de *func-sql-render*.

```
; render-page :: string string string -> expr?
; Construye página web de una tabla en específico.
; Retorna expresión.
(define (render-page sch npg ntb)
  (define css (string-append app-dir "/" sch "/htdocs/app.css"))
    `(html (head (title ,npg)
       (style ((type "text/css")) ,(file->string css #:mode 'text))
       (script ((type "text/javascript") (src "/app.js")) "␣"))
         (body ((onload "init();"))
           (div ((id "div-menu") (name "div-menu"))
             ,(menu-lst lsttables))
           (div ((id "div-form-buttons") (name "div-form-buttons"))
             ,@(form-sql-render npg ntb))
           (div ((id "msgrest") (name "msgrest")) "␣")
           (form ((id "form-real")) ,@(form-render-real ntb))
           ,(response-msg)
           ,@(no-form-sql-render npg ntb))))
```

La función *render-page* construye una nueva página web a través de la expresividad de Racket y las funciones menu-lst, form-sql-render, form-render-real, response-msg y no-form-sql-render. El procedimiento *menu-lst* permite desplegar el menu de las páginas creadas, *form-sql-render* crea los formularios web de acuerdo a las funciones CRUD especificadas, *form-render-real* construye los campos de la página de acuerdo a la configuración de campos especificada, *response-msg* construye un bloque para dar manejo a los mensajes de error y *no-form-sql-render* crea un bloque para dar manejo a las llamadas de funciones CRUD de tipo SELECT sin argumentos.

Para ver el resultado producido por la herramienta, especificamente el de las funciones previamente descritas, ligadas al proceso de generación de código fuente de las páginas, es necesario disponer de alguna configuración de campos y de funciones CRUD, almacenadas en las estructuras de datos correspondientes. Las configuraciones almacenadas en este caso son para la tabla *ventas* del esquema *empven*.

Nombre de campo	**Etiqueta**	**Tipo de elemento**	**Restricción**	**Tamaño Lógico**	**Tamaño Físico**	**Nulable**	**Mostrar**	**Lista de opciones**
totalventa	Total venta	text	numérico	10	10	NO	Si	Ninguno
fecventa	Fecha de venta	select_date	alfanumérico	10	10	NO	Si	Ninguno
numdoc	Número documento	text	numérico	10	10	NO	Si	Ninguno
id	ID venta	text	numérico	10	10	NO	Si	Ninguno

Tabla 5.2: Datos de configuración de campos - para tabla ventas

Nombre del procedimiento	**Tipo de procedimiento**	**Prototipo SQL**
select-all-venta	SELECT	SELECT totalventa, fecventa, numdoc, id FROM ventas
insert-venta	INSERT	INSERT totalventa, fecventa, numdoc FROM ventas

Tabla 5.3: Datos de configuración de funciones CRUD - para tabla ventas

Una vez se tengan las configuraciones establecidas y se inicie el proceso de generación de código fuente, el resultado que se obtiene a partir de estas funciones es el siguiente:

Código Fuente 5.12: Código fuente generado por procedimiento make-page

```
(define (empven-ventas args request)
 (local ((define (response-generator embed/url)
  (response/xexpr (quasiquote
  (html (head (title "empven-ventas") (style ((type "text/css")) "Contenido␣CSS.")
    (script ((type "text/javascript") (src "/app.js")) "␣"))
  (body ((onload "init();"))
   (div ((id "div-menu") (name "div-menu"))
    (ul (li (a ((href "/ventas")) "ventas"))))
   (div ((id "div-form-buttons") (name "div-form-buttons"))
     (form ((id "form-insert-venta") (class "insert-form")
       (method "post") (action (unquote (embed/url insert-venta))))
       (input ((type "hidden") (name "totalventa_crud") (id "totalventa_crud")))
       (input ((type "hidden") (name "fecventa_crud") (id "fecventa_crud")))
       (input ((type "hidden") (name "numdoc_crud") (id "numdoc_crud")))
       (input ((type "button") (value "insert-venta")
       (onclick "process_crud_form('form-insert-venta');")))))
       (div ((id "msgrest-container")) (div ((id "msgrest") (name "msgrest")) "␣"))

    (form ((id "form-real"))
     (div ((class "numerico"))
      (span ((class "titulo")) "Total␣venta")
       (input ((type "text") (name "totalventa") (id "totalventa") (size "10")
        (maxlength "10") (onblur "checkRestrictions(this);"))))
     (div ((class "alfanumerico"))
      (span ((class "titulo")) "Fecha␣de␣venta")
       (div ((class "date") (name "fecventa") (id "fecventa"))
        (select ((id "select_day"))
         "Contenido␣de␣elemento␣select_date")))
     (div ((class "numerico"))
      (span ((class "titulo")) "Numero␣documento")
       (input ((type "text") (name "numdoc") (id "numdoc") (size "10")
        (maxlength "10") (onblur "checkRestrictions(this);"))))
     (div ((class "numerico"))
      (span ((class "titulo")) "ID␣venta")
       (input ((type "text") (name "id") (id "id") (size "10")
        (maxlength "10") (onblur "checkRestrictions(this);")))))

     (unquote (if (string? args)
        (quasiquote (div ((id "message")) (span "*␣") (unquote args)))
        (if (list? args) (render-table "div-table-response" args) "")))

     (unquote (render-table "div-tableselect-all" (bd-select-all-venta))))))))

        (define (insert-venta request)
                (define bind (request-bindings request))
                (define totalventa (extract-binding/single (quote totalventa_crud) bind))
                (define fecventa (extract-binding/single (quote fecventa_crud) bind))
                (define numdoc (extract-binding/single (quote numdoc_crud) bind))
                (define nargs (bd-insert-venta totalventa fecventa numdoc))
                (empven-ventas nargs request)))
        (send/suspend/dispatch response-generator)))
```

Los pasos involucrados en el proceso de generación de una aplicación Web prototipo, se ilustran en el siguiente driagrama de flujo.

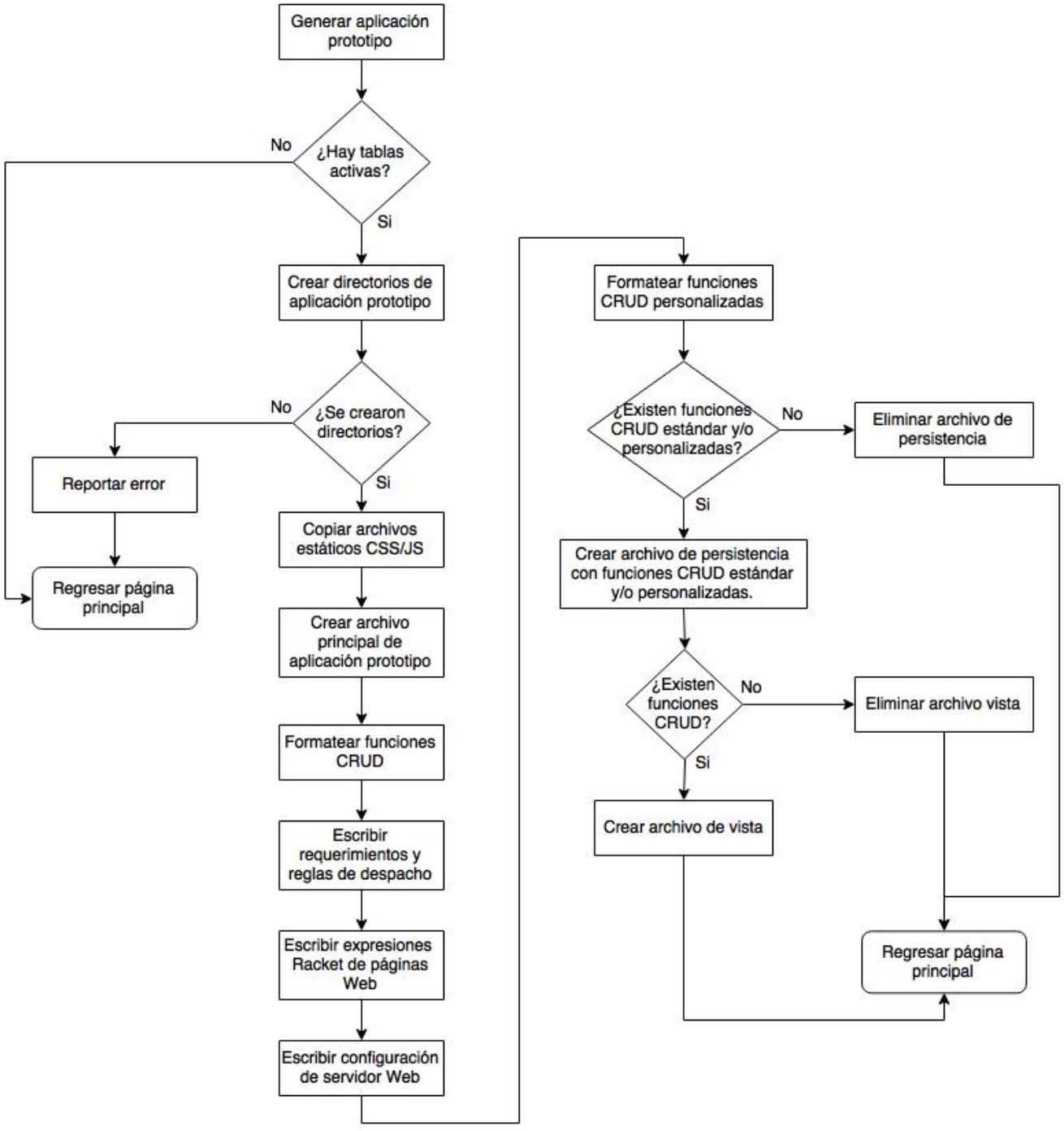


Figura 5.2: Diagrama de flujo - Generación de aplicación Web prototipo

Capítulo 6

Pruebas

6.1. Pruebas

Para comprobar el correcto funcionamiento de la herramienta, asegurar el cumplimiento de los requerimientos y calificar el grado de satisfacción del usuario se han elaborado un conjunto de pruebas. El tipo de pruebas evaluadas en el software son de aceptación, usabilidad y comparación.

6.1.1. Pruebas de aceptación

A través de estas pruebas se comprueba el cumplimiento de los requerimientos de la herramienta. Cada prueba tiene asociado mútiples casos de prueba, con el propósito de considerar y dar manejo correcto a las posibles situaciones que puede generar el usuario en el uso de la herramienta. Cada prueba se ha especificado de la siguiente manera:

- **Identificador**, asignación que identifica de forma única la prueba.
- **Referencia**, especifica a que requerimiento se encuentra relacionado la prueba.
- **Nombre**, especifica el nombre de la prueba.
- **Responsable**, especifica quien es el responsable de la prueba.
- **Estado**, especifica estado de la prueba. Los estados posibles son sin inspeccionar o inspeccionado.

Se describe a continuación la prueba PR-1 Registrar usuario.

Identificador	PR-1
Referencia	RF-1 Registrar usuario
Nombre	Registrar usuario
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

Cada prueba tiene asociado un conjunto de casos de pruebas. Cada uno direcciona la prueba que se realiza y los resultados que deben obtenerse. La información que describe cada caso de prueba es la siguiente:

- **Identificador**, asignación que identifica de forma única el caso de prueba.
- **Identificador de prueba**, especifica a que prueba pertenece.
- **Nombre**, especifica el nombre y tipo de prueba.
- **Usuario**, especifica el usuario que realiza la prueba.
- **Fecha de creación**, especifica la fecha en que se hizo la prueba.
- **Descripción**, especifica el propósito de la prueba.
- **Dependencias**, lista casos de prueba de los que depende.
- **Paso**, especifica el orden en que las acciones del usuario deben ser ejecutadas.
- **Usuario**, especifica las acciones que debe realizar el usuario.
- **Sistema**, especifica las acciones que realiza la herramienta.
- **Resultado esperado**, especifica el resultado que debe entregar la herramienta.

Se hicieron 16 pruebas, cada una con sus respectivos casos de prueba que oscilan entre 1 y 5, para un total de 31 casos de prueba. Se describe a continuación los siguientes casos de prueba.

- Caso de prueba CPR-1 de PR-1 Registrar usuario - Datos válidos
- Caso de prueba CPR-2 de PR-1 Registrar usuario - Omisión de campo
- Caso de prueba CPR-3 de PR-1 Registrar usuario - Datos inválidos
- Caso de prueba CPR-4 de PR-1 Registrar usuario - Usuario registrado
- Caso de prueba CPR-1 de PR-5 Editar tabla dinámica de configuración de campos - Datos válidos
- Caso de prueba CPR-2 de PR-5 Editar tabla dinámica de configuración de campos - Datos inválidos

- **CPR-1 PR-1 Registrar usuario - Datos válidos**

Identificador	CPR-1
Identificador de prueba	PR-1
Nombre	Registrar usuario - Datos válidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema permite el registro de un nuevo usuario especificado correctamente.
Dependencias	N/A

Paso	1
Usuario	Especifique los siguientes valores en el formulario de la página registro: Nombre: Esteban Apellidos: Duque Correo electrónico: ed@correo.com Sexo: Masculino Contraseña: eee555&&& Confirmar contraseña: eee555&&&
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione el evento “Registrar”
Sistema	El sistema valida autenticidad de los datos y registra el nuevo usuario.
Resultado esperado	El sistema registra al nuevo usuario y despliega mensaje “Consulta procesada con éxito. Verifique los datos.”.

Tabla 6.4: Caso de prueba CPR-1 PR-1 Registrar usuario - Datos válidos

- **CPR-2 PR-1 Registrar usuario - Omisión de un campo**

Identificador	CPR-2
Identificador de prueba	PR-1
Nombre	Registrar usuario - Omisión de un campo
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema valida la autenticidad de la información suministrada en el proceso de registro.
Dependencias	N/A

Paso	1
Usuario	Especifique los siguientes valores en el formulario de la página registro: Nombre: [vacío] Apellidos: Correa Correo electrónico: ec@correo.com Sexo: Masculino Contraseña: eee555&&& Confirmar contraseña: eee555&&&
Sistema	Sistema valida autenticidad de datos suministrados en cada campo.
Resultado esperado	El sistema despliega mensaje de error “Campo -Nombre- debe ser especificado.”.

Paso	2
Usuario	Accione el evento “Registrar”
Sistema	El sistema valida autenticidad de los datos suministrados en cada campo.
Resultado esperado	El sistema despliega mensaje de error “Campo -Nombre- debe ser especificado.”. El sistema cancela proceso de registro.

Tabla 6.7: Caso de prueba CPR-2 PR-1 Registrar usuario - Omisión de un campo

- **CPR-3 PR-1 Registrar usuario - Datos inválidos**

Identificador	CPR-3
Identificador de prueba	PR-1
Nombre	Registrar usuario - Datos inválidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema valida la autenticidad de la información suministrada en el proceso de registro.
Dependencias	N/A

Paso	1
Usuario	Especifique los siguientes valores en el formulario de la página registro: Nombre: nombre123 Apellidos: apellido456 Correo electrónico: correo.com Sexo: [sin especificar] Contraseña: 123 Confirmar contraseña: 123
Sistema	Sistema valida autenticidad de datos suministrados en cada campo.
Resultado esperado	El sistema despliega mensaje de error “Sólo se permiten caracteres del alfabeto”, “Sólo se permiten caracteres del alfabeto.”, “Especifique un correo electrónico.”, “Campo -Sexo- debe ser especificado.”, “Contraseña debe ser al menos de 8 caracteres que combine números, letras y simbolos como (!, &). Sin espacios.”, “Contraseña no aceptada.”. (Uno a la vez).

Paso	2
Usuario	Accione el evento “Registrar”
Sistema	El sistema valida autenticidad de los datos suministrados en cada campo.
Resultado esperado	El sistema despliega mensaje de error “Sólo se permiten caracteres del alfabeto.”. El sistema cancela proceso de registro.

Tabla 6.10: Caso de prueba CPR-3 PR-1 Registrar usuario - Datos inválidos

- **CPR-4 PR-1 Registrar usuario - Usuario registrado**

Identificador	CPR-4
Identificador de prueba	PR-1
Nombre	Registrar usuario - Usuario registrado
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema comprueba la existencia de un usuario.
Dependencias	Caso de prueba CPR-1 de prueba PR-1

Paso	1
Usuario	Especifique los siguientes valores en el formulario de la página registro: Nombre: Esteban Apellidos: Duque Correo electrónico: ed@correo.com Sexo: Masculino Contraseña: eee555&&& Confirmar contraseña: eee555&&&
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione el evento “Registrar”
Sistema	El sistema valida autenticidad y comprueba existencia de los datos suministrados.
Resultado esperado	El sistema despliega mensaje de error “Datos ya registrados.”.

Tabla 6.13: Caso de prueba CPR-4 PR-1 Registrar usuario - Usuario registrado

CPR-1 PR-5 Editar tabla dinámica de config. de campos - Datos válidos

Identificador	CPR-1
Identificador de prueba	PR-5
Nombre	Editar tabla dinámica de config. de campos - Datos válidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema permite la edición de las propiedades de los campos de una tabla del esquema de base de datos ingresando información válida a cada propiedad.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Resultado esperado	El sistema despliega tabla de configuración de campos.

Paso	2
Usuario	En la tabla dinámica de config. de campos desplegada, ingrese los siguientes valores válidos para los campos “nombre” e “idciudad”: * Nombre - Nombre de etiqueta: Nombre de empleado - Tipo de elemento: text - Restricción: alfabético - Tamaño físico: 25 - Mostrar: si - Lista de opciones: n/a * Idciudad - Nombre de etiqueta: Ciudad - Tipo de elemento: select - Restricción: alfanumérico - Tamaño físico: 15 - Mostrar: si - Lista de opciones: [“CIU1” “PAL”],[“CIU2” “CAL”],[“CIU3” “TUL”]
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Accione evento “Guardar” por cada propiedad especificada sobre algún campo. En este caso para los campos “nombre” e “iciudad”.
Sistema	El sistema valida autenticidad de los valores especificados.
Resultado esperado	El sistema establece el valor especificado y deja la celda como texto plano.

Paso	4
Usuario	Accione evento “Guardar”.
Sistema	El sistema almacena información temporalmente.
Resultado esperado	El sistema establece/modifica estructura de datos (lista) en la que almacena las configuraciones establecidas en la tabla de configuración de campos.

Paso	5
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Sistema	El sistema despliega tabla de configuración de campos.
Resultado esperado	El sistema despliega tabla de configuración de campos de tabla “empleado” mostrando información suministrada en el Paso 2.

Tabla 6.19: Caso de prueba CPR-1 PR-5 Editar tabla dinámica de configuración de campos - Datos válidos

CPR-2 PR-5 Editar tabla dinámica de config. de campos - Datos inválidos

Identificador	CPR-2
Identificador de prueba	PR-5
Nombre	Editar tabla dinámica de configuración de campos - Datos inválidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema permite la edición y comprobación de las propiedades de los campos de una tabla del esquema de base de datos ingresando información inválida a cada propiedad.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Resultado esperado	El sistema despliega tabla de configuración de campos.

Paso	2
Usuario	En la tabla dinámica de configuración de campos desplegada, ingrese los siguientes valores inválidos para los campos “nombre” e “idciudad”: * Nombre - Nombre de etiqueta: [Espacio] - Tipo de elemento: text - Restricción: alfabético - Tamaño físico: -10 - Mostrar: si - Lista de opciones: n/a * Idciudad - Nombre de etiqueta: Ciudad - Tipo de elemento: select - Restricción: alfanumérico - Tamaño físico: 15 - Mostrar: si - Lista de opciones: [“CIU1” “PAL”][],[]
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Accione evento “Guardar” por cada propiedad especificada sobre algún campo. En este caso para los campos “nombre” e “iciudad”.
Sistema	El sistema valida autenticidad de los valores especificados.
Resultado esperado	El sistema despliega mensaje de error “Especificación de campo es incorrecta.” No establece el valor y espera corrección o cancelación de la acción.

Paso	4
Usuario	Accione evento “Cancelar”.
Sistema	El sistema cancela acción de edición de propiedad de campo.
Resultado esperado	El sistema establece el valor anterior correcto y deja la celda como texto plano.

Tabla 6.24: Caso de prueba CPR-2 PR-5 Editar tabla dinámica de configuración de campos - Datos inválidos

6.1.2. Prueba comparativa

Con el fin de acercar al lector a las herramientas CASE generadoras de código fuente, conocer en mayor medida su estado y funcionamiento, y con ello hacer un contraste con la herramienta desarrollada. Se decide realizar una prueba comparativa sobre un único escenario, que permita valorar el proceso realizado.

Para la prueba se utilizaron las herramientas Code on Time, ASPRunner.NET y Anauj. Se usa como fuente de información el modelo relacional *empven* descrito en el capítulo de implementación. La prueba está enfocada en manifestar los aspectos existentes en el proceso de generación de código en cada herramienta, y no pretende favorecer una herramienta sobre otra. Debido a que la valoración está sujeta a la percepción del autor, no debe tomarse como concluyentes las determinaciones aquí mencionadas.

La prueba comparativa estuvo principalmente enfocada en la determinación de la existencia o no de los siguientes aspectos.

	Code on Time	ASPRunner.NET	Anauj
Generación automática de operaciones principales (crear, editar, buscar)	X	X	
Modificación de elementos en el lugar o en conjunto		X	X
Disposición de elementos gráficos y de filtrado	X	X	X
Consulta a base de datos como fuente de información	X	X	
Configuración guiada	X	X	X
Documentación externa	X	X	
Código fuente generado es indentado	X	X	
Código fuente dividido en módulos y/o paquetes	X	X	X
Interfaz gráfica estructurada	X	X	X

Tabla 6.25: Tabla comparativa de herramientas CASE según cada aspecto

Para una mejor comprensión de los aspectos mencionados, se describe cada uno según lo experimentado en cada herramienta.

Code on Time

- Establece por defecto la creación de tres escenarios por cada tabla de la base de datos. Los escenarios son: creación, edición y búsqueda de datos.
- La modificación de los escenarios se realiza en un espacio independiente y único para cada campo de cada tabla.
- La representación de cada campo y filtrado es extensa en elementos bajo nombres y tipos específicos. Permite realizar consultas en la base de datos como fuente de información de los campos.
- El proceso de configuración realizado para la generación de código fuente es guiado y documentado.
- El código fuente generado es indentado, organizado en paquetes y con diseño orientado a objetos.
- Los elementos de la interfaz gráfica se presentan en forma organizada y consistente.

ASPRunner.NET

- Establece por defecto cinco escenarios principales: creación, edición, búsqueda, importación y exportación de datos. Pueden agregarse hasta doce, cada uno con una función específica.
- La modificación de los escenarios se hace en un espacio independiente, en forma secuencial antes de la generación del código fuente y/o de forma gráfica en conjunto.
- La representación para cada campo se establece con elementos HTML reconocibles y con filtrado pre-establecido para diferentes tipos (p. ej. numérico, email, fecha, tarjeta de crédito, etc.). Permite el uso de consultas en la base de datos como fuente de información de los campos.
- El proceso de configuración no es guiado pero existe documentación.
- El código fuente generado es indentado, organizado en paquetes y con diseño orientado a objetos.
- Los elementos de la interfaz gráfica se presentan en forma organizada y consistente.

Anauj

- No establece ningún escenario por defecto, deben establecerse de forma manual por cada tabla.
- La modificación de los escenarios se hace en el lugar, no se reserva un espacio externo y/o diferente para cada campo.
- La representación de cada campo se establece con elementos HTML y con filtrado pre-establecido para diferentes tipos (p. ej. numérico, alfabético, alfanumérico, fecha, email, booleano, etc.). No permite realizar consultas en la base de datos como fuente de información de los campos.
- El proceso de configuración es guiado y escasamente documentado.
- El código fuente generado no es indentado, si organizado en módulos.
- Los elementos de la interfaz gráfica se presentan en forma organizada y consistente.

Como apreciación final, aunque en las tres herramientas se depende extrictamente de sus lenguajes de programación ASP.NET y Racket respectivamente, este último, sin embargo, permite su instalación y configuración en diferentes sistemas operativos (p. ej. Windows, GNU/Linux), otorgando portabilidad y facilidad en la integración con otras herramientas.

6.1.3. Prueba de usabilidad

Para la prueba de usabilidad se decide realizar una encuesta, cuyo objetivo es conocer la percepción y grado de satisfacción del usuario con respecto a las funcionalidades de la herramienta, así como sus ideas de mejora. La encuesta se ha realizado a una muestra de selección intencionada que incluye docentes, egresados y estudiantes de Ingeniería en sistemas y Tecnología en sistemas de información de la Universidad del Valle.

Para el desarrollo de la encuesta se le solicitó a los usuarios interactuar con la herramienta, siguiendo una serie de videos tutoriales que guiaba de principio a fin el proceso necesario para la generación de una aplicación Web prototipo, mostrando simultáneamente las características de esta. Posteriormente, se le solicitaba al usuario resolver una encuesta compuesta de 15 preguntas, 13 de selección múltiple y 2 abiertas. Para el desarrollo de estas dos actividades el usuario solo requería de un equipo de cómputo con conexión a Internet y acceso a un navegador web.

A fin de obtener una calificación objetiva y una especificación rigurosa, se decide elegir como público objetivo, personas afines con conocimientos en el área de sistemas y desarrollo de software principalmente. El perfil de las personas encuestadas por consiguiente es: docencia universitaria, docencia en secundaria, estudiantes de noveno y décimo semestre de Ingeniería en sistemas, analistas y/o desarrolladores de software.

La encuesta fue realizada entre los meses Diciembre del 2015 y Febrero del 2016. Hubo un total de 21 encuestados. Se presenta a continuación el contenido de la encuesta y los resultados obtenidos.

1. Una vez revisado los videos y realizado el proceso sugerido, lo que experimentó para generar una aplicación prototipo diría que es:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Muy Rápido	5	1	4.8 %	5
Rápido	4	13	61.9 %	52
Normal	3	4	19 %	12
Lento	2	3	14.3 %	6
Muy lento	1	0	0 %	0
			Calificación promedio	75/21 = 3.57

2. Una vez revisado los videos y realizado el proceso sugerido, diría usted que este es útil para generar una aplicación prototipo:

	Votos	**Porcentaje votos**
Si	21	100 %
No	0	0 %

3. Una vez revisado los videos y seguido el proceso sugerido, diría usted que este es transparente como usuario final, teniendo en cuenta que son macros los que están generando código a partir de un lenguaje de programación en el paradigma funcional.

	Votos	**Porcentaje votos**
Si	20	95.2 %
No	1	4.8 %

4. Los mecanismos que utilizó para generar una aplicación prototipo son:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Muy fáciles de manejar	5	3	14.3 %	15
Fáciles de manejar	3.75	8	38.1 %	30
Normales de manejar	2.50	6	28.6 %	15
Difíciles de manejar	1.25	4	19 %	5
			Calificación promedio	65/21 = 3.09

5. La navegabilidad que siguió en la herramienta para generar una aplicación prototipo es:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Muy fácil de seguir	5	4	19 %	20
Fácil de seguir	3.75	11	52.4 %	41.25
Normal de seguir	2.50	5	23.8 %	12.5
Difícil de seguir	1.25	1	4.8 %	1.25
			Calificación promedio	75/21 = 3.57

6. Como calificaría la ayuda aportada por la herramienta:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Excelente	5	3	14.3 %	15
Sobresaliente	4	11	52.4 %	44
Buena	3	4	19 %	12
Regular	2	3	14.3 %	6
Mala	1	0	0 %	0
			Calificación promedio	77/21 = 3.66

7. Esta de acuerdo con que el diseño de la interfaz gráfica es:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Excelente	5	0	0 %	0
Sobresaliente	4	7	33.3 %	28
Bueno	3	7	33.3 %	21
Aceptable	2	5	23.8 %	10
Insuficiente	1	2	9.5 %	2
			Calificación promedio	61/21 = 2.90

8. Qué proceso o procesos se le dificultó más en entender y aplicar:

	Votos	**Porcentaje votos**
Configuración de campos	3	9 %
Configuración de funciones CRUD	14	41 %
Almacenamiento de configuración de campos	5	15 %
Almacenamiento de funciones CRUD	7	20 %
Visualización de configuración de campos	1	3 %
Exportar aplicación	1	3 %
Ejecutar aplicación	3	9 %

9. Está de acuerdo con el mecanismo de edición de la tabla de configuración de campos.

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Completamente de acuerdo	5	5	23.8 %	25
En mayor parte de acuerdo	4	8	38.1 %	32
En buena parte de acuerdo	3	4	19 %	12
Poco de acuerdo	2	3	14.3 %	6
Nada de acuerdo	1	1	4.8 %	1
			Calificación promedio	76/21 = 3.61

10. Está de acuerdo con el mecanismo de edición de funciones CRUD (crear, leer, actualizar y eliminar).

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Completamente de acuerdo	5	7	33.3 %	35
En mayor parte de acuerdo	4	6	28.6 %	24
En buena parte de acuerdo	3	6	28.6 %	18
Poco de acuerdo	2	2	9.5 %	4
Nada de acuerdo	1	0	0 %	0
			Calificación promedio	81/21 = 3.85

11. Ha tenido experiencia con alguna aplicación similar a la herramienta que aquí experimentó.

	Votos	**Porcentaje votos**
Si	8	38.1 %
No	13	61.9 %

12. Justifica el uso de la herramienta para obtener los resultados que ella genera.

	Votos	**Porcentaje votos**
Si	19	90.5 %
No	2	9.5 %

13. Califique en un porcentaje entre 0 % y 100 % su grado de satisfacción en términos generales, sobre el uso de la aplicación, siendo 0 % nada satisfecho y 100 % completamente satisfecho.

Rango de satisfacción	**Votos**	**Porcentaje votos**	**Total grado satisfacción**
0	0	0 %	0
10	0	0 %	0
20	0	0 %	0
30	0	0 %	0
40	2	9.5 %	80
50	1	4.8 %	50
60	1	4.8 %	60
70	3	14.3 %	210
80	8	38.1 %	640
90	6	28.6 %	540
100	0	0 %	0
		Promedio grado satisfacción	1580/21 = 75.23

14. Presentó algún problema en el uso de la herramienta, si fue así, podría especificarlo. Algunas de las respuestas obtenidas fueron: (Ver en anexos la totalidad de las respuestas).

- Ninguno.
- No presenté ningún problema con la herramienta.
- Para un usuario final que no tenga ningún tipo de experiencia con este tipo de aplicaciones sería ideal colocar terminología común o general. ej. “función de persistencia” podría cambiar a “Almacenamiento”.
- No pude ingresar las opciones del iddpto y idciudad.
- Al momento de editar la propiedad “mostrar” el campo numdoc de la tabla empleados noté que no se guardaban los cambios entre visible y no visible, para esto se requirió hacer uso de las opciones save y guardar una precedida de la otra. De esta manera reconoció los cambios.

15. Como valoramos su experiencia con la herramienta, y queremos que esta sea lo mejor posible, por favor déjenos su comentario, crítica constructiva y/o noción de cambio. Algunas de las respuestas obtenidas fueron: (Ver en anexos la totalidad de las respuestas).

- Tal vez mejoraría la parte de diseño de la herramienta. Pero a nivel funcional me parece que es muy completa y muy útil.
- Mejorar los mensajes dentro de la aplicación.
- Existen herramientas mbd mucho mas prácticas y más visuales para el usuario final, siendo esta una tendencia en programación.

- A nivel general, si el usuario es desarrollador se entiende la lógica de lo que se debe hacer, pero para alguien que no programe va a tener dificultades. Hay una gran oportunidad de mejoría en cuanto a la interfaz de usuario y su usabilidad, debería ser transparente por ejemplo en opciones tener que hacer cosas como '[“llave” “valor”],[“” “”],..'. Pueden mejorar las ayudas siendo un poco más descriptivas y en lo posible mostrar ejemplos. No es claro que se tenga que dar click en los campos para modificar su configuración.

- En el proceso de configuración de campos se podría agilizar guardando automáticamente cada configuración o (pulsando enter), la especificación de la lista de opciones no es agradable y se congestiona mucho por las comas, comillas y corchetes. Estoy de acuerdo con la configuración de funciones CRUD personalizada en lugar de la normal. El proceso de almacenamiento para ambos procesos debería ser en un solo paso. Finalmente las ayudas son poco atractivas y se dificulta su lectura.

6.1.4. Conclusiones de la prueba

Una vez obtenidos los resultados de la encuesta, es posible apreciar diferentes puntos de vista de diversos usuarios, que amplian y/o profundizan la visión de la herramienta, resaltando o por el contrario dando a conocer puntos de flaqueza que pueden ser mejorados. Algunos puntos a tener en cuenta son:

- La percepción lenta de la aplicación tiene que ver esencialmente con acciones repetitivas y paso a paso que el usuario enfrenta en el proceso de configuración de campos y funciones CRUD. Es necesario un cambio en el diseño que elimine las acciones intermedias y/o las automatice, agilizando la experiencia del usuario.

- La presencia de ayudas pasivas, que no guian al usuario en el momento indicado y la inexistencia de una interfaz gráfica que revele u otorgue indicaciones sobre su forma de uso, hacen que las funcionalidades de la herramienta en un primer intento no queden claras y sean dificiles de implementar.

- El mecanismo actual de doble paso para el almacenamiento de configuración de campos y funciones CRUD, exige el constante accionar de dos eventos (cuatro en total) en áreas diferentes de la página, perjudicando la navegabilidad en simplicidad y tiempo.

- Más del 80 % de los encuestados está de acuerdo con el mecanismo de edición de la tabla de configuración de campos, lo que favorece su aceptación, pero no en forma definitiva. Con base en las sugerencias obtenidas es posible:

  - Adecuar el proceso de especificación de opciones, evitando la restricción impuesta por el formato actual.
  - Exigir la especificación de opciones para componentes que lo requieren.
  - Ilustrar con mayor precisión los errores que se cometen en el proceso de edición de cada tabla.
  - Agilizar el proceso de especificación de los valores para los campos.

- Más del 90 % de los encuestados apoya el mecanismo de edición de funciones CRUD, lo que consolida en forma definitiva su estructura, y posibilita la adición de caracteristicas como:

- Selección en un paso de todos los campos de la tabla para incluirlos en la función de persistencia en la configuración de funciones CRUD normal.
- Presentación constante y cercana de los campos de la tabla en la configuración personalizada.

Debido a que la encuesta tiene como propósito evaluar y conocer la viabilidad de la herramienta CASE en su actual estado de desarrollo, según su función y forma de uso, las sugerencias o cambios que sean requeridos no se harán de forma inmediata. No obstante, es necesario a futuro la planificación y realización de los cambios que aquí se recomiendan, con el propósito de dar solidez a su funcionalidad y usabilidad, y así mejorar significativamente la experiencia del usuario.

6.1.5. Ficha técnica

Solicitada por:	Víctor Andrés Bucheli Guerrero
Realizada por:	Héctor Mauricio Cabrera
Nombre de la encuesta:	Prueba de usabilidad - Herramienta CASE Anauj
Universo	Miembros del grupo de investigación EVALAB activos de la Universidad del Valle - Sede Melendez, Estudiantes de Ingenieria en Sistemas de la Universidad del Valle - Sede Tuluá, Egresados de Tecnología en Sistemas de Información de la Universidad del Valle - Sede Palmira.
Tipo de muestreo:	No probabilístico por selección intencionada.
Tamaño de la muestra:	21 encuestas, con margen de error de +/- 5 % para un nivel de confianza de 75 %, p=0.5 y q=0.5.
Fecha de creación:	3 de Diciembre del 2015
Técnica de recolección de datos:	Encuesta online alojada en portal Web.
Objetivo de la encuesta:	Conocer la percepción y grado de satisfacción del usuario con respecto a las funcionalidades de la herramienta, así como sus ideas de mejora.
Número de preguntas:	15
Tipo de preguntas aplicadas:	Cerradas (13) y Abiertas (2)
Escala empleada para medición:	Puntuación y semántica.

Tabla 6.26: Ficha técnica - Encuesta de prueba de usabilidad

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

1. La culminación del proyecto hace manifiesto el alcance del lenguaje de programación Racket, que a través del paradigma funcional permite no solo el desarrollo de aplicaciones Web, sino la entrega de un conjunto de herramientas (p. ej. servidor web, administración de cookies, expresividad, acceso de base de datos, etc.), que simplifican y homogeinízan su desarrollo.

2. El uso de la programación funcional promovió el desarrollo de la herramienta CASE, debido principalmente a la especificación directa y precisa de expresiones que siempre retornan un resultado, haciendo posible la composición de funciones y división de responsabilidades, útiles para el cumplimiento de los requerimientos.

3. El uso de las macros a través del lenguaje de programación Racket, ha hecho posible la creación de expresiones que se extienden y generan la definición de funciones de la aplicación prototipo, convirtiendo esta característica en un factor imprescindible en la generación de código fuente.

4. En la referencia consultada se encontró escasa especificación de funciones para la búsqueda, acceso y modificación de las estructuras de datos, lo que impuso la aplicación de un esfuerzo adicional en la construcción de dichas funciones, limitando el soporte asignado a la separación de los datos pertenecientes a cada usuario.

5. En la literatura revisada se notó ausencia de artefactos y/o instrumentos enfocados al diseño y documentación del desarrollo funcional (p. ej. contratos). Lo que limitó la especificación de los mecanismos internos de la herramienta, requiriendo la incorporación y adecuación de artefactos enfocados al diseño orientado a objetos, con el fin de lograr una descripción apropiada y comprensible del diseño y funcionamiento de la herramienta.

6. En el desarrollo de una herramienta CASE es importante la interacción del usuario final, que debe ser considerada desde muy temprano en el proceso de desarrollo y consolidada en la etapa de diseño, de manera que sea un componente clave, capaz de orientar oportunamente el proceso que debe seguir el usuario en la herramienta.

7. En la prueba de usabilidad se observa que la mayoría está de acuerdo en que el proceso realizado para generar una aplicación prototipo es rápido, aunque no es una respuesta definitiva, si es lo suficiente para validar su forma y alentar un trabajo posterior sobre ella.

8. En la prueba de usabilidad se determinó que el mecanismo de configuración de campos no presenta una aceptación convincente, debido a la difícil captación en su forma de uso, inflexible definición de opciones, falta de dinamismo e imprecisión en la comunicación de errores.

9. En la prueba de usabilidad se encontró que el mecanismo de configuración de funciones CRUD es ampliamente aceptado en el grupo de estudio, por lo que consolida en forma definitiva su estructura, y posibilita la adición de nuevas características que lo complementen.

10. La prueba comparativa aunque no es concluyente, demuestra la cercanía en algunos aspectos del proyecto con respecto a dos herramientas establecidas en el mercado (p. ej. modificación y disposición de elementos, configuración guiada, interfaz gráfica estructurada, etc.), destacando adicionalmente la libertad de poder ser configurada en diversos sistemas operativos (p. ej. Windows, GNU/Linux), una limitación evidente en las otras herramientas analizadas.

11. La prueba comparativa determinó las deficiencias presentes en el proyecto (p. ej. Generación automática de operaciones de consulta y almacenamiento, consulta a base de datos como fuente de información, código fuente generado legible, etc.), lo que dificulta su adopción en un medio productivo y advierte de los cambios necesarios.

12. El llevar a cabo la propuesta es la consideración y acercamiento de un pensamiento específico que rodea la programación funcional al desarrollo de aplicaciones, no solo en los términos necesarios para concebir un generador de código fuente sino con un propósito general. La propuesta desarrollada es un prototipo que dista de ser una herramienta CASE generadora de código fuente completa, que carece de una variedad de características esenciales, sin embargo, es una iniciativa que aporta una experiencia adecuada para considerar la construcción de un producto de software CASE, capaz de generar aplicaciones Web prototipo a través de la programación funcional.

7.2. Trabajos futuros

1. Es necesario un estudio y remodelamiento del diseño de las estructuras de datos, que permitan la separación y distinción de la información perteneciente a cada usuario, garantizando acceso y manipulación simultánea por múltiples usuarios en la herramienta. Se recomienda la combinación de listas y estructuras debido al poco esfuerzo involucrado en su manipulación y capacidad de extensión.

2. Una vez garantizado el acceso multiusuario a la herramienta, se recomienda la adopción de un diseño e implementación multihilo, debido principalmente a la alta carga de datos y procesamiento involucrado en sus procesos internos.

3. Se recomienda crear funciones de administración de base de datos que otorguen continuidad al proceso de generación de aplicaciones Web prototipo iniciado por los usuarios. Alguna de ellas es la creación y eliminación de esquemas de base datos, posible para cada usuario, cuyo resultado sea una experiencia completa e ininterrumpida.

4. Puesto que se trata de una herramienta de asistencia, debe ser claro para el usuario final el propósito de la herramienta y el procedimiento que sigue para lograrlo. Con respecto al procedimiento es posible: agilizar y clarificar la edición de la tabla de configuración de campos, presentar de manera oportuna mensajes de ayuda que guien el proceso, especificar con mayor detalle los errores cometidos, advertir sobre cada acción emprendida y reducir a una única acción cada mecanismo de almacenamiento.

5. La recolección y presentación inicial de los datos del usuario no es la más conveniente, no se sigue un estándar como XML o JSON, su estructura y funciones de acceso no son genéricas y además poco extensibles. La adopción de uno de estos estándares facilitaría la representación de los datos, así como su acceso y modificación de los mismos.

6. Para garantizar la generación de código legible y motivar su modificación manualmente, es preciso estudiar un nuevo mecanismo de impresión del código o el desarrollo de un programa externo, capaz de identar y ordenar el código fuente generado, acorde a la estructura común con que se especifica en el lenguaje Racket.

Bibliografía

[1] Dooley. John, *Software Development and Professional Practice*, Apress, 2011.

[2] Muller. Robert J., *Oracle Developer Starter Kit*, McGraw-Hill Professional Publishing, 1999.

[3] Muller. Robert J., *Oracle Developer Starter Kit*, McGraw-Hill Professional Publishing, 1999.

[4] Sommerville. Ian, *Ingeniería del software*, Pearson Addison Wesley, 2005.

[5] González López. Pascual, González López. Ana Amelia, Gallud Lázaro. José Antonio, "Herramientas CASE. ¿Cómo incorporarlas con éxito en nuestra organización?", *Ensayos: Revista de la Facultad de Educación de Albacete*, N° 10, pp. 195-208, 1995.

[6] Felleisen. Matthias, "Racket is ...", *Thoughts* [online]. Estados Unidos: Northeastern University, 2011. Disponible en: http://www.ccs.neu.edu/home/matthias/Thoughts/Racket_is____.html

[7] Graham. Paul, *On Lisp Advanced Techniques for Common Lisp*, Prentice Hall, 1993.

[8] Sub-Jefatura de Informática, *Herramientas CASE*, Instituto Nacional de Estadística e Informática, 1999.

[9] Hervás Lucas. Ramón, *Herramientas y Entornos de Programación* [online]. España: Universidad de Castilla-La Mancha, 2012. Disponible en: http://mami.uclm.es/rhervas/images/downloads/HyEP-Tema2-TecCASE-Part1Low.pdf

[10] Silberschatz. Abraham, Korth. Henry F., Sudarshan. S., *Fundamentos de bases de datos*, McGraw-Hill, 2002.

[11] Kroenke. David. M, *Procesamiento de bases de datos. Fundamentos, diseño e implementación*, Pearson Education, 2002.

[12] Bobrowski. Steve, *Oracle8i para Linux Edición de aprendizaje*, McGraw-Hill, 2001.

[13] Amelot. Michèle, *VBA Access 2007: Programar en Access*, Ediciones ENI, 2008.

[14] Tare. Ramkrishna S, *Procesamiento de datos en UNIX con INFORMIX-SQL, ESQL/C, C-ISAM y TURBO*, McGraw-Hill, 1990.

[15] Herrington. Jack, *Code Generation in Action*, Manning Publications, 2003.

[16] Flatt. Matthew, "Creating Languages in Racket", *acmqueue* [online], vol. 9, N° 11, 2011. Disponible en: http://queue.acm.org/detail.cfm?id=2068896

[17] Boardman. Susan, Melanie. Caffrey, Solomon. Morse, Benjamin. Rosenzweig, *Oracle Web Application Programming for PL/SQL Developers*, Prentice Hall Professional Technical Reference, 2003.

[18] Bass. Len, Clements. Paul, Kazman. Rick, *Software Architecture in Practice Second Edition*, Addison-Wesley Professional, 2003.

[19] Weyns. Danny, *Architecture-Based Design of Multi-Agent Systems*, Springer, 2010.

[20] Florez F. Héctor Arturo, "Arquitectura multicapa mediante AJAX y ORM", *Vínculos*, vol. 7, N°. 1, pp. 3-16, Junio 2010.

[21] Meza. David Alberto, "Sistema generador de aplicaciones Web multicapa - CODEV", Magíster en Tecnologías de Información, Departamento de Ciencias de la Computación, Universidad de Chile, Santiago de Chile, Junio 2008.

Anexos

Manual de instalación

Antes de poner en marcha la herramienta CASE es necesario cumplir con los siguientes requerimientos.

- Instalación y configuración del gestor de base de datos.
- Instalación de aplicativo DrRacket.

7.2.1. Instalación y configuración del gestor de base de datos

El gestor de base de datos elegido para el desarrollo del proyecto fue Postgresql 9.1. La instalación y configuración en un equipo con sistema operativo Debian GNU/Linux 7.9 ha sido efectuada a través de las siguientes instrucciónes.

```
$ sudo apt-get install postgresql postgresql-client
```

Una vez instalado el gestor de base de datos se procede a cambiar la contraseña del usuario postgres.

```
$ su -
# su postgres
# psql
# alter user postgres with pass '56648865';
```

Hecho esto se crea la base de datos codegen y los esquemas empven y estmat.

```
$ su -
# su postgres
# createdb -O postgres codegen
# psql -d codegen -U postgres
psql> \i codegen.sql
psql> \i empven.sql
psql> \i estmat.sql
```

7.2.2. Instalación de aplicativo DrRacket

La instalación del lenguaje de programación Racket y su entorno de desarrollo DrRacket se logra a través de la siguiente instrucción.

```
$ sudo apt-get install racket
```

Manual de usuario

Una vez finalizado el proceso de instalación se procede con el despliegue y uso de la herramienta. Para la puesta en marcha abra una terminal de comandos, desplácese hasta el directorio donde reside el archivo codegen.rkt y ejecute la siguiente instrucción: (En este caso el archivo se encuentra en el directorio /home/hmc/codegen-arq/recoleccion-datos).

```
$ racket -t codegen.rkt
```

Una vez se inicie el servidor, acceda a un navegador e ingrese a la dirección localhost:8000. Sabrá que el servidor ha iniciado cuando en el terminal aparezca el siguiente mensaje.

```
hmc@pdp7:~/codegen-arq/recoleccion-datos$ racket -t codegen.rkt
Your Web application is running at http://localhost:8000.
Stop this program at any time to terminate the Web Server.
```

El directorio donde reside la herramienta no debe ser igual al aquí expuesto, puede elegirlo en base a sus necesidades.

Proceso de registro

Para el proceso de registro diríjase desde la página de inicio de sesión a la página de registro a través del enlace, o especifique en el navegador la dirección localhost:8000/register. Una vez ahí especifique todos los datos solicitados y de click en registrar. Si ocurre algún error o si el registro es exitoso será informado. Una vez hecho el registro puede regresar a la página de inicio de sesión a través del enlace. Ver Figura 7.1.

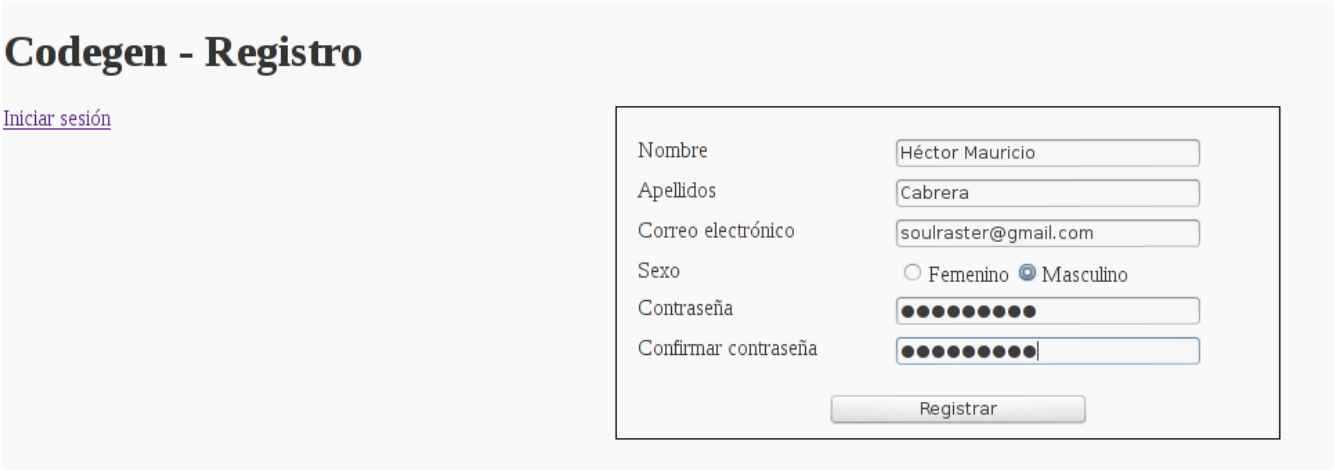


Figura 7.1: Proceso de registro

Proceso de inicio de sesión

Si efectúo el registro con éxito, puede iniciar sesión especificando su correo electrónico, contraseña y esquema de base de datos asociado. Ver Figura 7.2.

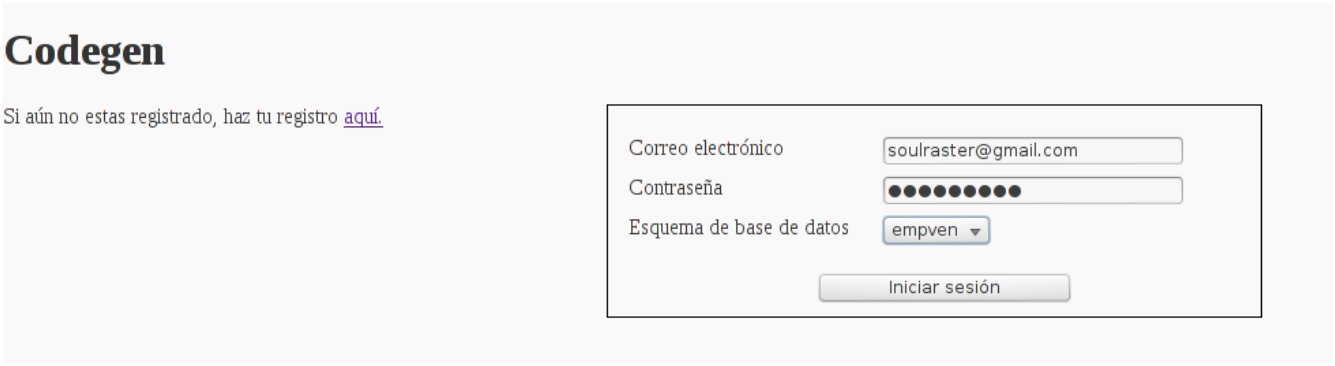


Figura 7.2: Proceso de inicio de sesión

Configuración de campos

La herramienta por defecto selecciona siempre la primera tabla del esquema para mostrar la tabla de configuración de campos, tablas de funciones CRUD y formularios de configuración de funciones CRUD. Una vez haya seleccionado la tabla del esquema que desea configurar, al costado derecho se desplegará la tabla dinámica de configuración de campos. Para iniciar la modificación de los campos basta con dar click en la celda que se intersecta entre la columna de alguna propiedad y la fila de algún campo.

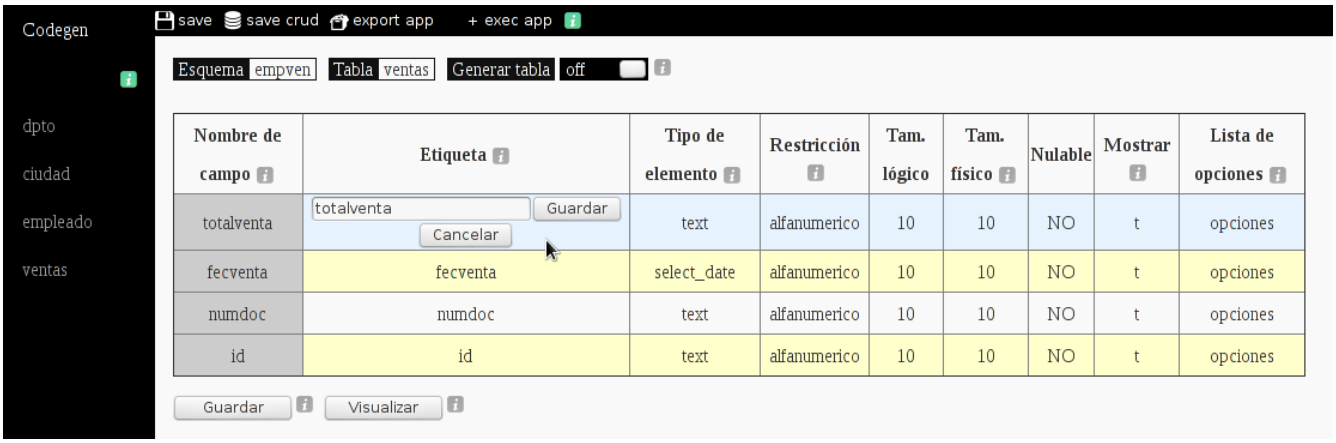


Figura 7.3: Configuración de campos

Cada modificación de un campo que origine debe ser guardado o cancelado. Si cancela, el valor anterior correcto será establecido. Ver Figura 7.3. Si requiere alguna información de como debe establecer la propiedad de algún campo, de click en el símbolo de información del campo, en la parte superior de la tabla al lado de cada nombre de propiedad. Ver Figura 7.4.

Una vez haya configurado los campos proceda a almacenarlos temporalmente. Para eso de click en el botón 'Guardar' ubicado en la parte inferior de la tabla. Habrá guardado la configuración cuando la página en que se encuentra se recargue. El almacenamiento temporal tendrá las configuraciones de los campos mientras este activo en la herramienta. Puede sentirse libre de pasar a otra tabla y hacer las

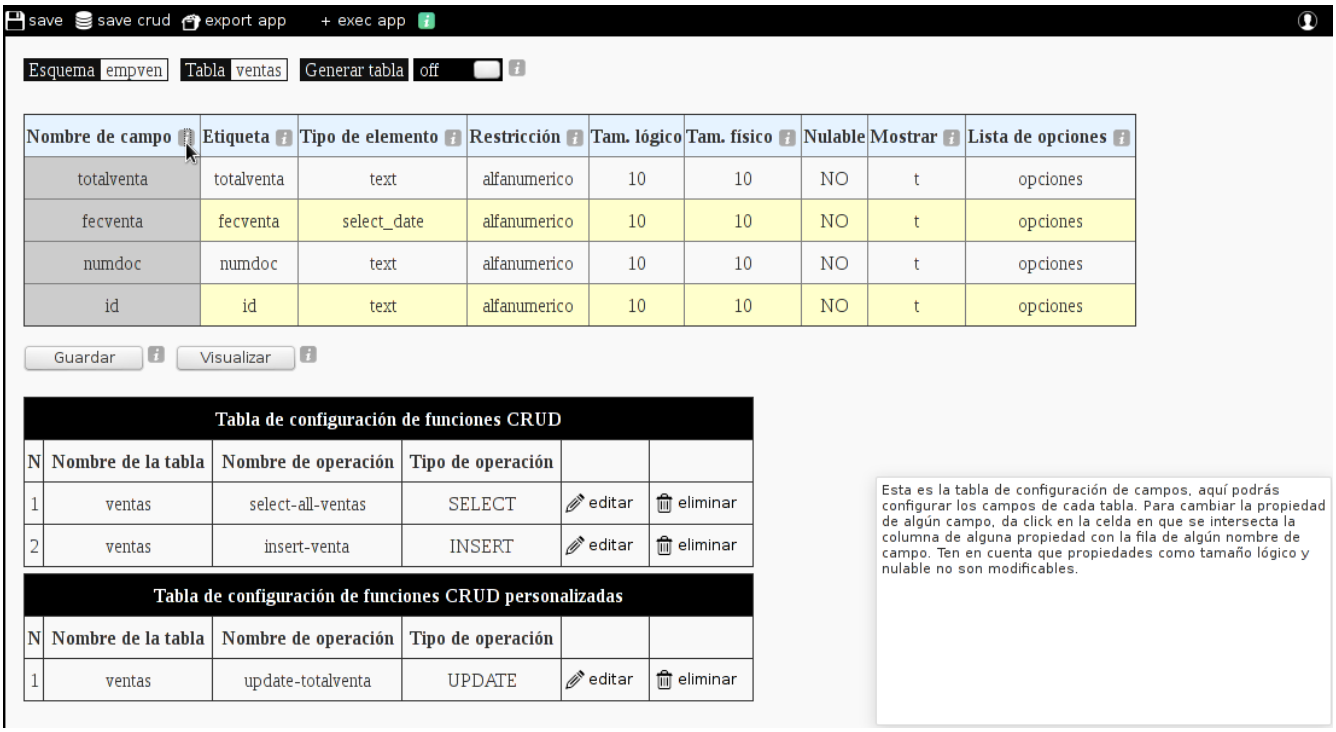


Figura 7.4: Mensajes de información

configuraciones necesarias. Una vez considere que estén hechas puede almacenar todas las configuraciones de forma permanente a través de 'save' en la parte superior de la herramienta. Esto permitirá que una vez abandone la herramienta las configuraciones se conserven. Ver Figura 7.5.

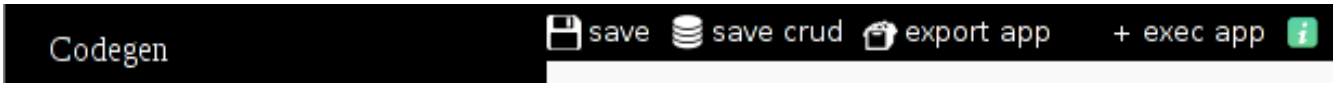


Figura 7.5: Menú principal de la herramienta

Visualización de configuración de campos

Si desea observar gráficamente la configuración hecha a los campos, de click en el botón 'Visualizar' en la parte inferior de la tabla dinámica de configuración de campos. Esto le permitirá visualizar la configuración de los campos de la tabla actual. Tenga presente que las configuraciones de los campos debieron haber sido almacenados temporalmente. Ver Figura 7.6.

Configuración de funciones CRUD

Para la configuración de las funciones CRUD existen dos mecanismos: La configuración estándar y la configuración personalizada. Para la configuración estándar diríjase a la sección "Función SQL" y

Figura 7.6: Visualización de configuración de campos

seleccione la función que desea establecer, hecho esto la herramienta despliega el formulario que debe especificar para la creación de una nueva función CRUD. Es necesario que al menos una función CRUD sea creada. Ver Figura 7.7.

Si requiere ayuda de como especificar un campo del formulario, cada uno posee el símbolo de información, de click en el. Una vez haya establecido los valores de la función CRUD puede almacenarla temporalmente a través del botón 'Guardar' en la parte inferior del formulario. Si lo hace, en la parte superior del formulario será listada la función que acaba de crear. La tabla que es creada, es la tabla de funciones CRUD, en ella verá listadas todas las funciones CRUD estándar que fueron creadas y están asociadas a la tabla del esquema actual. A través de ella podrá modificar o eliminar una función CRUD. Ver Figura 7.8.

Para la configuración de funciones CRUD personalizadas diríjase a la sección "Función SQL personalizada" y de click en el botón próximo. Se desplegará un formulario, en el que solo deberá especificar dos campos, nombre del procedimiento y definición de instrucción SQL. Ver Figura 7.9. La opción 'Usar update para invalidar' será necesaria solo en casos especiales. A excepción de la función SQL INSERT todas las demás funciones (SELECT, UPDATE y DELETE) deben tener restricciones, es decir ir acompañadas de la cláusula WHERE. Para comprender la especificación de las funciones SQL puede utilizar los mensajes de información asociados a cada campo. Una vez haya especificado la función SQL personalizada debe guardarla de forma temporal a través del botón 'Guardar', ubicado en la parte inferior del formulario. Esto al igual que la configuración de funciones CRUD estándar, listará en una nueva tabla la función creada en la parte superior. A través de ella puede editar y eliminar cualquier función CRUD personalizada que haya creado. Ver Figura 7.8.

A diferencia de la configuración de campos, cuando haya creado/editado/eliminado las funciones CRUD estándar y personalizadas de una tabla, deberá almacenarlas permanentemente en archivo de forma inmediata antes de hacer cualquier otra cosa. Esto se logra a través del evento 'save crud' ubicado en la parte superior de la herramienta. Ver Figura 7.5. Esto permitirá que una vez abandone la herramienta las configuraciones se conserven. Si no lo hace en este orden y pasa a otra página primero las configuraciones

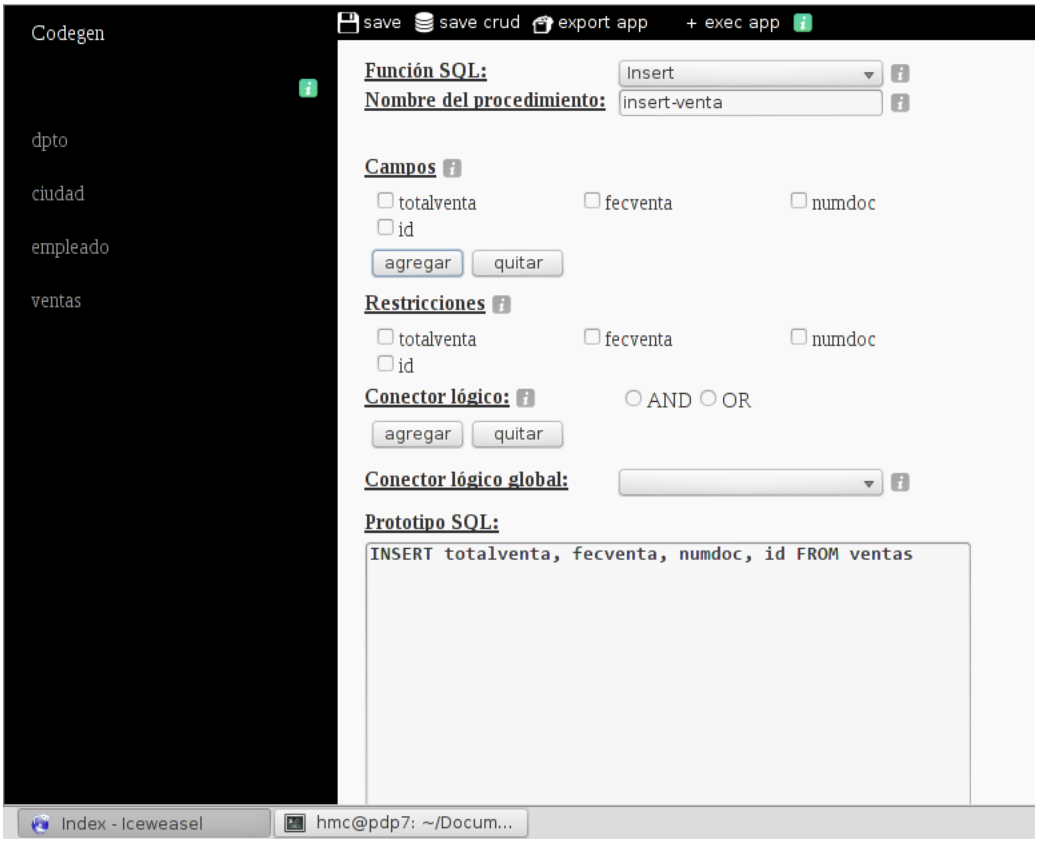


Figura 7.7: Configuración de funciones CRUD

Tabla de configuración de funciones CRUD

N	Nombre de la tabla	Nombre de operación	Tipo de operación		
1	ventas	select-all-ventas	SELECT	editar	eliminar
2	ventas	insert-venta	INSERT	editar	eliminar

Tabla de configuración de funciones CRUD personalizadas

N	Nombre de la tabla	Nombre de operación	Tipo de operación		
1	ventas	udpate-totalventa	UPDATE	editar	eliminar

Figura 7.8: Tabla de configuración de funciones CRUD estándar y personalizadas

Figura 7.9: Configuración de funciones CRUD personalizadas

creadas/editadas/eliminadas no serán consolidadas correctamente.

Activación de tablas

Para la generación del código fuente de la aplicación Web prototipo, es necesario especificar que tablas deben ser generadas. Para activar una tabla e incluirla en la generación del código fuente debe establecer en (on) el elemento de activación. Ver Figura 7.10.

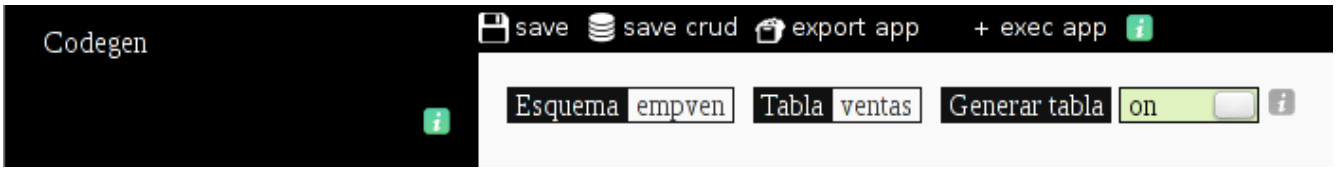


Figura 7.10: Activación de tablas

Debe tener en cuenta que cada vez que inicie sesión el elemento de activación estará desactivado (off), por lo que todas las tablas en principio siempre estarán inactivas.

Generación de aplicación

Cuando haya terminado de configurar los campos, funciones CRUD y activado las tablas necesarias, puede generar la aplicación Web prototipo a través de 'export app' ubicado en la parte superior de la herramienta. Esto creará los archivos de código fuente bajo el directorio *generacion_datos/apps/nombre_esquema*. Ver Figura 7.5.

Puesta en marcha de aplicación generada

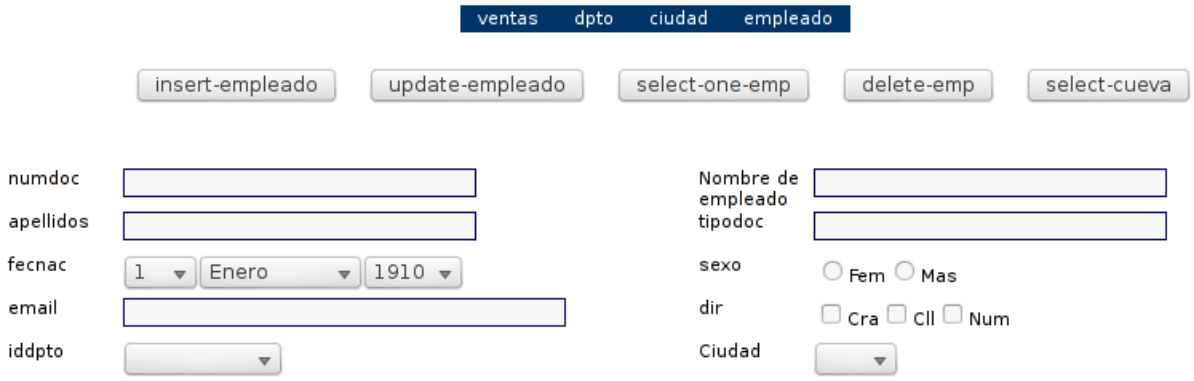
Para la puesta en marcha de la aplicación prototipo abra una terminal de comandos y diríjase al directorio de la aplicación ubicado en el directorio *generacion_datos/apps/nombre_esquema* a partir del directorio actual de la herramienta. Es decir la aplicación será generada en *directorio_herramienta/generacion_datos/apps/nombre_esquema*. En este caso sería.

```
hmc@pdp7:~/codegen-arq/recoleccion-datos$ cd ../generacion-datos/apps/empven
```

Una vez esté en el directorio *nombre_esquema* ejecute la aplicación con la siguiente instrucción: racket -t nombre_esquema-app.rkt. En este caso sería.

```
hmc@pdp7:~/codegen-arq/generacion-datos/apps/empven$ racket -t empven.rkt
```

Esto pondrá en ejecución la aplicación Web prototipo en la dirección URL http://localhost:8900, acceda a esta dirección desde el navegador para comprobar la ejecución de la aplicación Web prototipo. Ver Figura 7.11. La alternativa inmediata es accionar el evento 'exec app' en la parte superior de la herramienta. Ver Figura 7.5.



CIU1	DPTO1	dir	email	F	2015-3-10	cc	apee
CIU1	DPTO1	cra 23 # 40-21	new@gmail.com	F	2015-1-1	cc	newape
CIU1	DPTO1	cra 56 # 56	jc@correo.com	F	2015-4-3	cc	castaño
CIU1	DPTO1	dir	mc@gmail.com	F	2010-1-1	cc	contreras
CIU1	DPTO1	dir	jr@gmail.com	F	2010-1-1	cc	romero
CIU1	DPTO1	dir	er@correo.com	F	2010-1-1	cc	ruiz
CIU1	DPTO1	cll 5	linadq@correo.com	F	2012-5-4	cc	duque
CIU1	DPTO1	Num	so@correo.com	M	2010-12-3	cc	orozco
CIU1	DPTO1	Num	as@correo.com	M	2012-4-4	cc	solarte

Figura 7.11: Puesta en marcha de aplicación Web prototipo

Anexos

Anauj, Herramienta CASE generadora de código fuente de aplicaciones Web prototipo

Héctor Mauricio Cabrera Murillo Mejía

Índice general

Capítulo 1

Requerimientos

1.1. Requerimientos funcionales

1.1.1. RF-1

Identificador	RF-1
Nombre del requerimiento	Registrar usuario
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	15/08/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	22/08/2015
Resumen	El usuario ingresa información personal exigida para el registro, hecho esto el sistema valida, registra y notifica al usuario.
Entradas	Información personal del usuario (Correo electrónico, nombre, apellidos, fecha de nacimiento, sexo, contraseña).
Resultados	Registro en el sistema del nuevo usuario.
Curso Básico de eventos	1. El usuario ingresa información personal exigida para el registro. 2. El sistema valida autenticidad de la información suministrada. 3. El sistema registra el nuevo usuario. 4. El sistema notifica al usuario.
Caminos alternativos	N/A

Caminos de excepción	1. Punto 2. Si usuario omite algún campo, el sistema despliega un mensaje de error indicando que hace falta especificar información. 2. Punto 2. Si usuario especifica incorrectamente un campo (caracteres no válidos), el sistema despliega un mensaje de error indicando la especificación incorrecta del campo. 3. Punto 3. Si el usuario ya se encuentra registrado, el sistema notifica que los datos ya se encuentran registrados.
Pre - condiciones	N/A
Post - condiciones	N/A

Cuadro 1.1: RF-1

1.1.2. RF-2

Identificador	RF-2
Nombre del requerimiento	Iniciar sesión
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	20/08/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	22/08/2015
Resumen	El usuario especifica información requerida para iniciar sesión, hecho esto el sistema valida, comprueba e ingresa al aplicativo.
Entradas	El usuario especifica información de sesión (Correo electrónico, contraseña y esquema de base de datos).
Resultados	El sistema inicia sesión y permite acceso del usuario al aplicativo.
Curso Básico de eventos	1. El usuario especifica información requerida para iniciar sesión. 2. El sistema valida información suministrada. 3. El sistema comprueba existencia de usuario. 4. El sistema ingresa al aplicativo.
Caminos alternativos	N/A
Caminos de excepción	1. Punto 2. Si usuario omite algún campo, el sistema despliega mensaje de error indicando la falta de información. 2. Punto 3. Si correo electrónico y/o contraseña del usuario son incorrectos, el sistema despliega mensaje de error indicando la inconsistencia. 3. Punto 3. Si esquema de base de datos especificado no se encuentra asociado al usuario (relacionado en la base de datos) el sistema despliega mensaje de error indicando la inconsistencia. 4. Punto 3. Si esquema de base de datos especificado no se encuentra creado en la base de datos, el sistema despliega mensaje de error indicando la inconsistencia.
Pre - condiciones	Debe existir usuario registrado en el sistema.
Post - condiciones	N/A

Cuadro 1.2: RF-2

1.1.3. RF-3

Identificador	RF-3
Nombre del requerimiento	Cerrar sesión
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	24/08/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	26/08/2015
Resumen	Usuario acciona evento salir del aplicativo, hecho esto el sistema desvincula al usuario del aplicativo. (Usuario debe iniciar sesión de nuevo para ingresar al aplicativo).
Entradas	Ninguna.
Resultados	El sistema desvincula al usuario del aplicativo.
Curso Básico de eventos	1. Usuario acciona evento salir del aplicativo.
Caminos alternativos	N/A
Caminos de excepción	Ninguno.
Pre - condiciones	Usuario debe haber iniciado sesión.
Post - condiciones	Información de sesión del usuario almacenada temporalmente en (cookies) es eliminada.

Cuadro 1.3: RF-3

1.1.4. RF-4

Identificador	RF-4
Nombre del requerimiento	Desplegar elementos del esquema de base de datos
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	23/03/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	20/08/2015
Resumen	Hecho el proceso de inicio de sesión, el sistema despliega los elementos que conforman el esquema de base de datos.
Entradas	N/A
Resultados	El sistema despliega elementos del esquema de base de datos (campos y tablas), seleccionado en el inicio de sesión del usuario.
Curso Básico de eventos	1. Despliegue de nombres de las tablas del esquema de base de datos. 2. Despligue de campos de las tablas del esquema de base de datos.
Caminos alternativos	N/A
Caminos de excepción	N/A
Pre - condiciones	El usuario debe haber iniciado sesión.
Post - condiciones	N/A

Cuadro 1.4: RF-4

1.1.5. RF-5

Identificador	RF-5
Nombre del requerimiento	Desplegar tabla dinámica de configuración de campos
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	28/03/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	20/08/2015
Resumen	El usuario selecciona el nombre de la tabla del esquema de base de datos a configurar, hecho esto el sistema consulta y despliega la tabla de configuración de campos.
Entradas	Nombre de tabla que pertenece al esquema de la base de datos.
Resultados	Despliegue de tabla de configuración de campos.
Curso Básico de eventos	1. Usuario selecciona nombre de tabla del esquema de base de datos. 2. El sistema consulta los campos y propiedades (nombre, tipo de dato, tamaño lógico) de la tabla seleccionada. 3. El sistema agrega nuevas propiedades a los campos consultados (nombre de esquema, nombre de tabla, nombre de etiqueta, tipo de elemento, restricción, tamaño físico, nulable, mostrar, lista de opciones). 4. El sistema despliega tabla de configuración de campos conformado por cada campo y sus propiedades.
Caminos alternativos	1. Punto 2/Punto 3. Si existe archivo de configuracion de campos cargado en una estructura de datos (lista), el sistema consulta los campos y sus propiedades desde esta estructura y no desde la fuente inicial (base de datos).
Caminos de excepción	N/A
Pre - condiciones	El usuario debe haber iniciado sesión.
Post - condiciones	N/A

Cuadro 1.5: RF-5

1.1.6. RF-6

Identificador	RF-6
Nombre del requerimiento	Editar tabla dinámica de configuración de campos
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	11/04/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	21/08/2015
Resumen	El usuario especifica valores para las propiedades de los campos (nombre de etiqueta, tipo de elemento, restricción, tamaño físico, mostrar, lista de opciones) pertenecientes a la tabla seleccionada del esquema de base de datos y acciona evento guardar. El sistema valida y almacena temporalmente los valores de las propiedades de cada campo.
Entradas	Propiedades de los campos (nombre de esquema, nombre de tabla, nombre de campo, nombre de etiqueta, tipo de elemento, restricción, tamaño lógico, tamaño físico, nulable, mostrar, lista de opciones).
Resultados	Almacenamiento temporal de las propiedades de los campos en estructura de datos (lista).
Curso Básico de eventos	1. Usuario especifica valores a las propiedades de los campos. 2. El sistema valida autenticidad de los valores especificados. 3. Usuario acciona evento guardar. 4. El sistema almacena temporalmente valores de las propiedades de los campos.
Caminos alternativos	N/A
Caminos de excepción	1. Punto 2. Si valor de la propiedad del campo especificado es incorrecto (caracteres no válidos), el sistema despliega mensaje de error indicando la inconsistencia y cancela asignación del valor. (Se deja valor correcto anterior).
Pre - condiciones	El usuario debe haber iniciado sesión. El usuario debe haber seleccionado una tabla del esquema de la base de datos.
Post - condiciones	Almacenamiento temporal en estructura de datos (lista) de los valores de las propiedades de los campos.

Cuadro 1.6: RF-6

1.1.7. RF-7

Identificador	RF-7
Nombre del requerimiento	Almacenar lista de configuración de campos
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	27/04/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	21/08/2015
Resumen	Usuario acciona evento "save app" (guardar aplicación), el sistema almacena permanentemente la estructura de datos que conserva las propiedades de los campos de la totalidad de las tablas del esquema de base de datos.
Entradas	Estructura de datos (lista).
Resultados	Almacenamiento permanente (en archivo) de la estructura de datos (en lista) que conserva las propiedades de los campos de la totalidad de las tablas del esquema de base de datos.
Curso Básico de eventos	1. Usuario acciona evento "save app" (guardar aplicación). 2. Sistema almacena permanentemente las propiedades de los campos.
Caminos alternativos	N/A
Caminos de excepción	N/A
Pre - condiciones	El usuario debe haber iniciado sesión.
Post - condiciones	Almacenamiento permanente (en archivo) de las propiedades de los campos de la totalidad de las tablas del esquema de base de datos.

Cuadro 1.7: RF-7

1.1.8. RF-8

Identificador	RF-8
Nombre del requerimiento	Despliegue de tabla dinámica de funciones CRUD - estándar
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	12/05/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	25/08/2015
Resumen	El usuario selecciona el nombre de la tabla del esquema de base de datos, hecho esto el sistema consulta y despliega la tabla dinámica de funciones CRUD.
Entradas	Nombre de tabla que pertenece al esquema de base de datos.
Resultados	Tabla dinámica de funciones CRUD.
Curso Básico de eventos	1. Usuario selecciona tabla del esquema de base de datos. 2. Sistema consulta funciones CRUD creadas y las despliega en tabla dinámica.
Caminos alternativos	N/A
Caminos de excepción	1. Punto 2. Si existe archivo de configuración de funciones CRUD, el sistema consulta las funciones desde el archivo y las despliega en la tabla dinámica. 2. Punto 2. Si no existe archivo de configuración el sistema no despliega tabla dinámica. 3. Punto 2. Si no existe en el archivo relación de funciones CRUD con respecto a la tabla seleccionada, el sistema no despliega tabla dinámica.
Pre - condiciones	El usuario debe haber iniciado sesión.
Post - condiciones	N/A

Cuadro 1.8: RF-8

1.1.9. RF-9

Identificador	RF-9
Nombre del requerimiento	Editar formulario de configuración de funciones CRUD - estándar
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	26/06/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	25/08/2015
Resumen	Usuario especifica función CRUD a crear (insert, select, uptade, delete), nombre del procedimiento, campos y restricciones involucrados en la función. Posteriormente el usuario acciona el evento guardar. Hecho esto el sistema valida y almacena temporalmente la función CRUD especificada.
Entradas	Función CRUD a crear, nombre del procedimiento, campos y restricciones.
Resultados	Almacenamiento temporal de función CRUD en estructura de datos (lista).
Curso Básico de eventos	1. Usuario especifica función CRUD a crear, nombre del procedimiento, selecciona campos y restricciones necesarios en la función. 2. Usuario acciona evento guardar. 3. Sistema valida autenticidad de datos especificados y almacena temporalmente función CRUD.
Caminos alternativos	N/A
Caminos de excepción	1. Punto 3. Si usuario omite la especificación de la función CRUD, nombre del procedimiento o campos, el sistema no efectúa el almacenamiento y espera a que el usuario suministre información correcta.
Pre - condiciones	El usuario debe haber iniciado sesión. El usuario debe haber seleccionado una tabla del esquema de base de datos.
Post - condiciones	Almacenamiento temporal en estructura de datos (lista) de funciones CRUD.

Cuadro 1.9: RF-9

1.1.10. RF-10

Identificador	RF-10
Nombre del requerimiento	Almacenar lista de configuración de funciones CRUD - estándar
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	10/07/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	26/08/2015
Resumen	Usuario acciona evento "save crud" (guardar crud), el sistema almacena permanentemente la estructura de datos que conserva las funciones CRUD de la totalidad de las tablas del esquema de base de datos.
Entradas	Estructura de datos (lista).
Resultados	Almacenamiento permanente (en archivo) de la estructura de datos (en lista) que conserva las funciones CRUD de la totalidad de las tablas del esquema de base de datos.
Curso Básico de eventos	1. Usuario acciona evento "save crud" (guardar crud). 2. Sistema almacena permanentemente las funciones CRUD.
Caminos alternativos	N/A
Caminos de excepción	N/A
Pre - condiciones	El usuario debe haber iniciado sesión.
Post - condiciones	Almacenamiento permanente (en archivo) de las funciones CRUD de la totalidad de las tablas del esquema de base de datos.

Cuadro 1.10: RF-10

1.1.11. RF-11

Identificador	RF-11
Nombre del requerimiento	Despliegue de tabla dinámica de funciones CRUD - personalizada
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	12/11/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	24/11/2015
Resumen	El usuario selecciona el nombre de la tabla del esquema de base de datos, hecho esto el sistema consulta y despliega la tabla dinámica de funciones CRUD personalizadas.
Entradas	Nombre de tabla que pertenece al esquema de base de datos.
Resultados	Tabla dinámica de funciones CRUD personalizadas.
Curso Básico de eventos	1. Usuario selecciona tabla del esquema de base de datos. 2. Sistema consulta funciones CRUD personalizadas creadas y las despliega en tabla dinámica.
Caminos alternativos	N/A
Caminos de excepción	1. Punto 2. Si existe archivo de configuración de funciones CRUD personalizadas, el sistema consulta las funciones desde el archivo y las despliega en la tabla dinámica. 2. Punto 2. Si no existe archivo de configuración el sistema no despliega tabla dinámica. 3. Punto 2. Si no existe en el archivo relación de funciones CRUD personalizadas con respecto a la tabla seleccionada, el sistema no despliega tabla dinámica.
Pre - condiciones	El usuario debe haber iniciado sesión.
Post - condiciones	N/A

Cuadro 1.11: RF-11

1.1.12. RF-12

Identificador	RF-12
Nombre del requerimiento	Editar formulario de configuración de funciones CRUD - personalizada
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	18/11/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	26/11/2015
Resumen	El usuario especifica el nombre del procedimiento y la función CRUD a crear (Instrucción SQL insert, select, uptade o delete) en el formato exigido por el lenguaje de programación Racket. Posteriormente el usuario acciona el evento guardar. Hecho esto el sistema valida y almacena temporalmente la función CRUD especificada.
Entradas	Nombre del procedimiento y especificación completa de función CRUD a crear (Instrucción SQL insert, select, update o delete).
Resultados	Almacenamiento temporal de función CRUD personalizada en estructura de datos (lista).
Curso Básico de eventos	1. Usuario establece nombre del procedimiento y especificación completa de función CRUD a crear. 2. Usuario acciona evento guardar. 3. Sistema valida autenticidad de datos especificados y almacena temporalmente función CRUD.
Caminos alternativos	N/A
Caminos de excepción	1. Punto 3. Si usuario omite la especificación de la función CRUD, nombre del procedimiento o los especifica de forma incorrecta, el sistema no efectúa el almacenamiento y espera a que el usuario suministre información correcta.
Pre - condiciones	El usuario debe haber iniciado sesión. El usuario debe haber seleccionado una tabla del esquema de base de datos.
Post - condiciones	Almacenamiento temporal en estructura de datos (lista) de funciones CRUD personalizadas.

Cuadro 1.12: RF-12

1.1.13. RF-13

Identificador	RF-13
Nombre del requerimiento	Almacenar lista de configuración de funciones CRUD - personalizada
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	21/11/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	29/11/2015
Resumen	Usuario acciona evento "save crud" (guardar crud), el sistema almacena permanentemente la estructura de datos que conserva las funciones CRUD personalizadas de la totalidad de las tablas del esquema de base de datos.
Entradas	Estructura de datos (lista).
Resultados	Almacenamiento permanente (en archivo) de la estructura de datos (en lista) que conserva las funciones CRUD personalizadas de la totalidad de las tablas del esquema de base de datos.
Curso Básico de eventos	1. Usuario acciona evento "save crud" (guardar crud). 2. Sistema almacena permanentemente las funciones CRUD personalizadas.
Caminos alternativos	N/A
Caminos de excepción	N/A
Pre - condiciones	El usuario debe haber iniciado sesión.
Post - condiciones	Almacenamiento permanente (en archivo) de las funciones CRUD personalizadas de la totalidad de las tablas del esquema de base de datos.

Cuadro 1.13: RF-13

1.1.14. RF-14

Identificador	RF-14
Nombre del requerimiento	Visualizar gráficamente configuración de tabla dinámica de los campos
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	02/05/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	22/08/2015
Resumen	Usuario acciona evento visualizar, hecho esto el sistema consulta y despliega gráficamente la configuración de los campos.
Entradas	Estructura de datos (lista).
Resultados	Despliegue gráfico de las propiedades de los campos pertenecientes a la tabla seleccionada del esquema de base de datos.
Curso Básico de eventos	1. Usuario acciona evento visualizar. 2. Sistema despliega gráficamente las propiedades de los campos.
Caminos alternativos	N/A
Caminos de excepción	N/A
Pre - condiciones	Usuario debe haber iniciado sesión. Usuario debe haber seleccionado una tabla del esquema de base de datos.
Post - condiciones	N/A

Cuadro 1.14: RF-14

1.1.15. RF-15

Identificador	RF-15
Nombre del requerimiento	Exportar aplicación
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Alta
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	19/08/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	28/08/2015
Resumen	Usuario
Entradas	Estructura de datos (lista) de configuración de campos de la totalidad de tablas del esquema de base de datos. Estructura de datos (lista) de funciones CRUD (Opcional).
Resultados	Generación de archivos con código fuente del aplicativo.
Curso Básico de eventos	1. Usuario selecciona tablas del esquema de base de datos que son incluidas en la generación del código fuente del aplicativo. 2. Usuario acciona evento exportar aplicación. 3. Sistema consulta lista de configuración de campos, lista de configuración de funciones CRUD y genera aplicación prototipo.
Caminos alternativos	N/A
Caminos de excepción	1. Punto 3. Si usuario no selecciona ninguna tabla, el sistema no genera aplicación prototipo.
Pre - condiciones	Usuario debe haber iniciado sesión.
Post - condiciones	Generación de archivos con código fuente de aplicativo prototipo.

Cuadro 1.15: RF-15

1.1.16. RF-16

Identificador	RF-16
Nombre del requerimiento	Generar mensajes de información
Actor(es)	Usuario
Indispensable/Deseable	Indispensable
Prioridad	Media
Visible/No visible	Visible
Autor	Héctor Mauricio Cabrera
Fecha de elaboración	08/09/2015
Revisión	Héctor Mauricio Cabrera
Fecha de última revisión	12/09/2015
Resumen	El usuario acciona evento informar (icono), hecho esto el sistema despliega mensaje de información.
Entradas	N/A
Resultados	Despliegue de mensaje de información.
Curso Básico de eventos	1. Usuario acciona evento informar (icono). 2. Sistema despliega mensaje de información.
Caminos alternativos	N/A
Caminos de excepción	N/A
Pre - condiciones	Usuario debe haber iniciado sesión.
Post - condiciones	N/A

Cuadro 1.16: RF-16

1.2. Requerimientos no funcionales

1.2.1. RNF-1

Identificador	RNF-1
Nombre del requerimiento	Lenguaje de programación
Tipo	Restricción Tecnológica
Indispensable/Deseable	Indispensable
Crítico	Si
Descripción	El desarrollo del software asociado al proyecto debe estar realizado en los siguientes lenguajes de programación: Racket, HTML, CSS, JavaScript.
Criterios de aceptación	La Implementación del proyecto debe hacerse sobre estas tecnologías/lenguajes.

Cuadro 1.17: RNF-1

1.2.2. RNF-2

Identificador	RNF-2
Nombre del requerimiento	Sistema de gestión de base de datos
Tipo	Restricción Tecnológica
Indispensable/Deseable	Indispensable
Crítico	Si
Descripción	El desarrollo del software asociado al proyecto debe estar soportado en un sistema gestor de base de datos relacional. El sistema elegido es PostgreSQL.
Criterios de aceptación	La totalidad de la implementación del proyecto debe soportarse sobre este sistema de gestión de base de datos.

Cuadro 1.18: RNF-2

Capítulo 2

Pruebas

2.1. Pruebas de aceptación

2.1.1. PR-1

Identificador	PR-1
Referencia	RF-1 Registrar usuario
Nombre	Registrar usuario
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.2. PR-2

Identificador	PR-2
Referencia	RF-2 Iniciar sesión
Nombre	Iniciar sesión
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.3. PR-3

Identificador	PR-3
Referencia	RF-4 Desplegar elementos del esquema de base de datos
Nombre	Desplegar elementos del esquema de base de datos
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.4. PR-4

Identificador	PR-4
Referencia	RF-5 Desplegar tabla dinámica de configuración de campos
Nombre	Desplegar tabla dinámica de configuración de campos
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.5. PR-5

Identificador	PR-5
Referencia	RF-6 Editar tabla dinámica de configuración de campos
Nombre	Editar tabla dinámica de configuración de campos
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.6. PR-6

Identificador	PR-6
Referencia	RF-7 Almacenar lista de configuración de campos
Nombre	Almacenar lista de configuración de campos
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.7. PR-7

Identificador	PR-7
Referencia	RF-8 Desplegar tabla dinámica de funciones CRUD
Nombre	Desplegar tabla dinámica de funciones CRUD
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.8. PR-8

Identificador	PR-8
Referencia	RF-9 Editar formulario de configuración de funciones CRUD
Nombre	Editar formulario de configuración de funciones CRUD
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.9. PR-9

Identificador	PR-9
Referencia	RF-10 Almacenar lista de configuración de funciones CRUD
Nombre	Almacenar lista de configuración de funciones CRUD
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.10. PR-10

Identificador	PR-10
Referencia	RF-14 Visualizar gráficamente configuración de campos
Nombre	Visualizar gráficamente configuración de campos
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.11. PR-11

Identificador	PR-11
Referencia	RF-15 Exportar aplicación
Nombre	Exportar aplicación
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.12. PR-12

Identificador	PR-12
Referencia	RF-16 Generar mensajes de información
Nombre	Generar mensajes de información
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.13. PR-13

Identificador	PR-13
Referencia	RF-3 Cerrar sesión
Nombre	Cerrar sesión
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.14. PR-14

Identificador	PR-14
Referencia	RF-11 Desplegar tabla dinámica de funciones CRUD - personalizada
Nombre	Desplegar tabla dinámica de funciones CRUD personalizadas
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.15. PR-15

Identificador	PR-15
Referencia	RF-12 Editar formulario de configuración de funciones CRUD - personalizada
Nombre	Editar formulario de configuración de funciones CRUD personalizadas
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.1.16. PR-16

Identificador	PR-16
Referencia	RF-13 Almacenar lista de configuración de funciones CRUD - personalizada

Nombre	Almacenar lista de configuración de funciones CRUD personalizadas
Responsable	Héctor Mauricio Cabrera
Estado	Inspeccionado

2.2. Casos de pruebas

2.2.1. PR-1 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-1
Nombre	Registrar usuario - Datos válidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema permite el registro de un nuevo usuario especificado correctamente.
Dependencias	N/A

Paso	1
Usuario	Especifique los siguientes valores en el formulario de la página registro: Nombre: Esteban Apellidos: Duque Correo electrónico: ed@correo.com Sexo: Masculino Contraseña: eee555&&& Confirmar contraseña: eee555&&&
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione el evento “Registrar”
Sistema	El sistema valida autenticidad de los datos y registra el nuevo usuario.
Resultado esperado	El sistema registra al nuevo usuario y despliega mensaje “Consulta procesada con éxito. Verifique los datos.”.

2.2.2. PR-1 CPR-2

Identificador	CPR-2
Identificador de prueba	PR-1
Nombre	Registrar usuario - Omisión de un campo
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema valida la autenticidad de la información suministrada en el proceso de registro.
Dependencias	N/A

Paso	1
Usuario	Especifique los siguientes valores en el formulario de la página registro: Nombre: [vacío] Apellidos: Correa Correo electrónico: ec@correo.com Sexo: Masculino Contraseña: eee555&&& Confirmar contraseña: eee555&&&
Sistema	Sistema valida autenticidad de datos suministrados en cada campo.
Resultado esperado	El sistema despliega mensaje de error “Campo -Nombre- debe ser especificado.”.

Paso	2
Usuario	Accione el evento “Registrar”
Sistema	El sistema valida autenticidad de los datos suministrados en cada campo.
Resultado esperado	El sistema despliega mensaje de error “Campo -Nombre- debe ser especificado.”. El sistema cancela proceso de registro.

2.2.3. PR-1 CPR-3

Identificador	CPR-3
Identificador de prueba	PR-1
Nombre	Registrar usuario - Datos inválidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema valida la autenticidad de la información suministrada en el proceso de registro.
Dependencias	N/A

Paso	1
Usuario	Especifique los siguientes valores en el formulario de la página registro: Nombre: nombre123 Apellidos: apellido456 Correo electrónico: correo.com Sexo: [sin especificar] Contraseña: 123 Confirmar contraseña: 123
Sistema	Sistema valida autenticidad de datos suministrados en cada campo.
Resultado esperado	El sistema despliega mensaje de error “Sólo se permiten caracteres del alfabeto”, “Sólo se permiten caracteres del alfabeto.”, “Especifique un correo electrónico.”, “Campo -Sexo- debe ser especificado.”, “Contraseña debe ser al menos de 8 caracteres que combine números, letras y simbolos como (!, &). Sin espacios.”, “Contraseña no aceptada.”. (Uno a la vez).

Paso	2
Usuario	Accione el evento “Registrar”
Sistema	El sistema valida autenticidad de los datos suministrados en cada campo.
Resultado esperado	El sistema despliega mensaje de error “Sólo se permiten caracteres del alfabeto.”. El sistema cancela proceso de registro.

2.2.4. PR-1 CPR-4

Identificador	CPR-4
Identificador de prueba	PR-1
Nombre	Registrar usuario - Usuario registrado
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema comprueba la existencia de un usuario.
Dependencias	Caso de prueba CPR-1 de prueba PR-1

Paso	1
Usuario	Especifique los siguientes valores en el formulario de la página registro: Nombre: Esteban Apellidos: Duque Correo electrónico: ed@correo.com Sexo: Masculino Contraseña: eee555&&& Confirmar contraseña: eee555&&&
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione el evento “Registrar”
Sistema	El sistema valida autenticidad y comprueba existencia de los datos suministrados.
Resultado esperado	El sistema despliega mensaje de error “Datos ya registrados.”.

2.2.5. PR-2 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-2
Nombre	Iniciar sesión - Datos válidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema permite autenticación e inicio de sesión de un usuario registrado en la base de datos y relacionado con un esquema de base de datos.
Dependencias	Caso de prueba CPR-1 de prueba PR-1

Paso	1
Usuario	En el formulario de la página de inicio de sesión especifique los siguientes valores: - Correo electrónico: soulraster@gmail.com - Contraseña: hector - Esquema de base de datos: empven
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione evento “Iniciar sesión”.
Sistema	El sistema valida autenticidad de los datos, comprueba existencia de usuario e inicia sesión.
Resultado esperado	El sistema inicia sesión e ingresa a página principal del aplicativo. Se crean los archivos temporales de identificación del usuario. (cookies).

2.2.6. PR-2 CPR-2

Identificador	CPR-2
Identificador de prueba	PR-2
Nombre	Iniciar sesión - Omisión de campo
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema valida autenticidad de los valores especificados en el proceso de inicio de sesión.
Dependencias	Caso de prueba CPR-1 de prueba PR-1

Paso	1
Usuario	En el formulario de la página de inicio de sesión especifique los siguientes valores: - Correo electrónico: [vacío] - Contraseña: [vacío] - Esquema de base de datos: empven
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione evento “Iniciar sesión”.
Sistema	El sistema valida autenticidad de datos especificados.
Resultado esperado	El sistema despliega mensaje de error “Especifique un usuario”. El sistema cancela inicio de sesión.

2.2.7. PR-2 CPR-3

Identificador	CPR-3
Identificador de prueba	PR-2
Nombre	Iniciar sesión - Campo incorrecto
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema valida autenticidad de los valores especificados en el proceso de inicio de sesión.
Dependencias	Caso de prueba CPR-1 de prueba PR-1

Paso	1
Usuario	En el formulario de la página de inicio de sesión especifique los siguientes valores: - Correo electrónico: soulraster@gmail.com - Contraseña: abc - Esquema de base de datos: empven
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione evento “Iniciar sesión”.
Sistema	El sistema valida autenticidad de datos especificados.
Resultado esperado	El sistema despliega mensaje de error “Usuario y/o contraseña incorrectos.”. El sistema cancela inicio de sesión.

2.2.8. PR-2 CPR-4

Identificador	CPR-4
Identificador de prueba	PR-2
Nombre	Iniciar sesión - Esquema de base de datos no creado
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema valida autenticidad de los valores especificados en el proceso de inicio de sesión.
Dependencias	Caso de prueba CPR-1 de prueba PR-1 Nota: Esquema no se encuentra creado en la base de datos.

Paso	1
Usuario	En el formulario de la página de inicio de sesión especifique los siguientes valores: - Correo electrónico: soulraster@gmail.com - Contraseña: hector - Esquema de base de datos: invalido
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione evento “Iniciar sesión”.
Sistema	El sistema valida autenticidad de datos especificados.
Resultado esperado	El sistema despliega mensaje de error “Esquema no se encuentra creado y/o relacionado.”. El sistema cancela inicio de sesión.

2.2.9. PR-2 CPR-5

Identificador	CPR-5
Identificador de prueba	PR-2
Nombre	Iniciar sesión - Esquema de base de datos no relacionado
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema valida autenticidad de los valores especificados en el proceso de inicio de sesión.
Dependencias	Caso de prueba CPR-1 de prueba PR-1 Nota: Esquema se encuentra creado pero no asociado al usuario en la base de datos. (Relación en tabla proyecto).

Paso	1
Usuario	En el formulario de la página de inicio de sesión especifique los siguientes valores: - Correo electrónico: ed@correo.com - Contraseña: eee555&&& - Esquema de base de datos: empven
Sistema	N/A
Resultado esperado	N/A

Paso	2
Usuario	Accione evento “Iniciar sesión”.
Sistema	El sistema valida autenticidad de datos especificados.
Resultado esperado	El sistema despliega mensaje de error “Esquema no se encuentra creado y/o relacionado.”. El sistema cancela inicio de sesión.

2.2.10. PR-3 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-3
Nombre	Desplegar elementos del esquema de base de datos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar el despliegue por primera vez de los elementos de la base de datos (tablas y campos) una vez el usuario ha iniciado sesión.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Inicia sesión.
Sistema	El sistema consulta los elementos del esquema de base de datos (tablas y campos), selecciona la primera tabla del esquema, despliega los nombres de las tablas, tabla dinámica de configuración de campos y tabla de funciones CRUD de tabla seleccionada.
Resultado esperado	Despliegue de elementos del esquema de base de datos.

2.2.11. PR-4 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-4
Nombre	Desplegar tabla dinámica de configuración de campos - Fuente inicial (base de datos)
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega correctamente la tabla de configuración de campos.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos en la pagina principal.
Sistema	El sistema consulta campos y propiedades de la tabla seleccionada desde el esquema de la base de datos. El sistema adiciona propiedades adicionales con valores establecidos por defecto.
Resultado esperado	El sistema despliega tabla de configuración de campos con valores por defecto de tabla seleccionada.

2.2.12. PR-4 CPR-2

Identificador	CPR-2
Identificador de prueba	PR-4
Nombre	Desplegar tabla dinámica de configuración de campos - Fuente secundaria (archivo)
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega correctamente la tabla de configuración de campos con datos provenientes desde un archivo.
Dependencias	Caso de prueba CPR-1 de prueba PR-5 Caso de prueba CPR-1 de prueba PR-6

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos en la pagina principal.
Sistema	El sistema consulta campos y propiedades de la tabla seleccionada desde estructura de datos temporal (lista cargada desde archivo).
Resultado esperado	El sistema despliega tabla de configuración de campos con valores almacenados en estructura de datos temporal. Los valores de propiedades para los campos “nombre” e “id-ciudad” coinciden con los especificados en CPR-1 PR-5.

2.2.13. PR-5 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-5
Nombre	Editar tabla dinámica de configuración de campos - Datos válidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema permite la edición de las propiedades de los campos de una tabla del esquema de base de datos ingresando información válida a cada propiedad.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Resultado esperado	El sistema despliega tabla de configuración de campos.

Paso	2
Usuario	En la tabla dinámica de configuración de campos desplegada, ingrese los siguientes valores válidos para los campos “nombre” e “idciudad”: * Nombre - Nombre de etiqueta: Nombre de empleado - Tipo de elemento: text - Restricción: alfabetico - Tamaño físico: 25 - Mostrar: si - Lista de opciones: n/a * Idciudad - Nombre de etiqueta: Ciudad - Tipo de elemento: select - Restricción: alfanumerico - Tamaño físico: 15 - Mostrar: si - Lista de opciones: [“CIU1” “PAL”],[“CIU2” “CAL”],[“CIU3” “TUL”]

Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Accione evento “Guardar” por cada propiedad especificada sobre algún campo. En este caso para los campos “nombre” e “iciudad”.
Sistema	El sistema valida autenticidad de los valores especificados.
Resultado esperado	El sistema establece el valor especificado y deja la celda como texto plano.

Paso	4
Usuario	Accione evento “Guardar”.
Sistema	El sistema almacena información temporalmente.
Resultado esperado	El sistema establece/modifica estructura de datos (lista) en la que almacena las configuraciones establecidas en la tabla de configuración de campos.

Paso	5
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Sistema	El sistema despliega tabla de configuración de campos.
Resultado esperado	El sistema despliega tabla de configuración de campos de tabla “empleado” mostrando información suministrada en el Paso 2.

2.2.14. PR-5 CPR-2

Identificador	CPR-2
Identificador de prueba	PR-5
Nombre	Editar tabla dinámica de configuración de campos - Datos inválidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema permite la edición y comprobación de las propiedades de los campos de una tabla del esquema de base de datos ingresando información inválida a cada propiedad.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Resultado esperado	El sistema despliega tabla de configuración de campos.

Paso	2
Usuario	En la tabla dinámica de configuración de campos desplegada, ingrese los siguientes valores inválidos para los campos “nombre” e “idciudad”: * Nombre - Nombre de etiqueta: [Espacio] - Tipo de elemento: text - Restricción: alfabetico - Tamaño físico: -10 - Mostrar: si - Lista de opciones: n/a * Idciudad - Nombre de etiqueta: Ciudad - Tipo de elemento: select - Restricción: alfanumerico - Tamaño físico: 15 - Mostrar: si - Lista de opciones: [“CIU1” “PAL”][],[]
Sistema	N/A

Resultado esperado	N/A

Paso	3
Usuario	Accione evento “Guardar” por cada propiedad especificada sobre algún campo. En este caso para los campos “nombre” e “iciudad”.
Sistema	El sistema valida autenticidad de los valores especificados.
Resultado esperado	El sistema despliega mensaje de error “Especificación de campo es incorrecta.” No establece el valor y espera corrección o cancelación de la acción.

Paso	4
Usuario	Accione evento “cancelar”.
Sistema	El sistema cancela acción de edición de propiedad de campo.
Resultado esperado	El sistema establece el valor anterior correcto y deja la celda como texto plano.

2.2.15. PR-6 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-6
Nombre	Almacenar lista de configuración de campos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema almacena permanentemente (en archivo) los valores establecidos en la lista de configuración de campos.
Dependencias	Caso de prueba CPR-1 de prueba PR-2 Caso de prueba CPR-1 de prueba PR-5

Paso	1
Usuario	Usuario acciona evento “save app” (guardar aplicación).
Sistema	El sistema almacena lista de configuración y recarga página actual.
Resultado esperado	El sistema almacena permanentemente (en archivo) lista de configuración de campos de la totalidad de las tablas del esquema de base de datos. El sistema recarga página actual.

Paso	2
Usuario	Efectue los siguientes casos de prueba (Cerrar sesión) e (Iniciar sesión).
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Seleccione tabla “empleado”.
Sistema	El sistema despliega tabla de configuración de campos.
Resultado esperado	El sistema despliega tabla de configuración de campos de tabla “empleado”. Los valores de los campos desplegados coinciden con los especificados en el caso de prueba CPR-1 de prueba PR-5.

2.2.16. PR-7 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-7
Nombre	Desplegar tabla dinámica de funciones CRUD - Inexistencia de archivo
Usuario	Héctor Mauricio Cabrera
Fecha de creación	21/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega correctamente la tabla de configuración de funciones CRUD.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Sistema	El sistema no despliega tabla de configuración de funciones CRUD.
Resultado esperado	El sistema no despliega tabla de configuración de funciones CRUD debido a la no existencia de archivo de configuración.

2.2.17. PR-7 CPR-2

Identificador	CPR-2
Identificador de prueba	PR-7
Nombre	Desplegar tabla dinámica de funciones CRUD - Inexistencia de relación
Usuario	Héctor Mauricio Cabrera
Fecha de creación	21/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega correctamente la tabla de configuración de funciones CRUD.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Sistema	El sistema no despliega tabla de configuración de funciones CRUD.
Resultado esperado	El sistema no despliega tabla de configuración de funciones CRUD debido a la no existencia de valores relacionados con tabla “empleado”.

2.2.18. PR-7 CPR-3

Identificador	CPR-3
Identificador de prueba	PR-7
Nombre	Desplegar tabla dinámica de funciones CRUD - Existencia de archivo
Usuario	Héctor Mauricio Cabrera
Fecha de creación	21/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega correctamente la tabla de configuración de funciones CRUD.
Dependencias	Caso de prueba CPR-1 de prueba PR-8 Caso de prueba CPR-1 de prueba PR-9

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Sistema	El sistema despliega tabla de configuración de funciones CRUD.
Resultado esperado	El sistema consulta funciones CRUD pertenecientes a tabla “empleado” y despliega tabla de configuración.

2.2.19. PR-8 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-8
Nombre	Editar formulario de configuración de funciones CRUD - Datos válidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	21/09/2015
Descripción	Esta prueba tiene como objetivo verificar la creación de función CRUD a una tabla del esquema de base de datos.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado”.
Resultado esperado	El sistema despliega formulario de configuración de funciones CRUD.

Paso	2
Usuario	En el formulario de configuración de funciones CRUD especificar los siguientes valores: - Función: SELECT - Nombre de procedimiento: select-all-empleado - Campos: idciudad, iddpto, dir, email, sexo, fecnac, tipodoc, apellidos, nombre, numdoc - Restricciones: n/a - Operador lógico: n/a
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Accione el evento “Guardar”.
Sistema	El sistema valida autenticidad y registra valores especificados temporalmente.

Resultado esperado	El sistema establece/modifica estructura de datos (lista) en la que almacena la configuración de función CRUD especificada en paso 1. El sistema despliega tabla de configuración de funciones CRUD en la parte superior del formulario.

2.2.20. PR-8 CPR-2

Identificador	CPR-2
Identificador de prueba	PR-8
Nombre	Editar formulario de configuración de funciones CRUD - Datos inválidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	21/09/2015
Descripción	Esta prueba tiene como objetivo validar autenticidad de los datos suministrados en la creación de función CRUD.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado”.
Resultado esperado	El sistema despliega formulario de configuración de funciones CRUD.

Paso	2
Usuario	En el formulario de configuración de funciones CRUD especificar los siguientes valores: - Función: SELECT - Nombre de procedimiento: [vacío] - Campos: n/a - Restricciones: n/a - Operador lógico: n/a
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Accione el evento “Guardar”.
Sistema	El sistema valida autenticidad de datos suministrados.
Resultado esperado	El sistema detiene la creación de función CRUD, espera a que usuario especifique campos correctamente o cancele operación.

Paso	4
Usuario	Accione el evento “Cancelar”.
Sistema	El sistema cancela operación de creación de función CRUD.
Resultado esperado	El sistema restablece valores del formulario de creación de función CRUD.

2.2.21. PR-9 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-9
Nombre	Almacenar lista de configuración de funciones CRUD
Usuario	Héctor Mauricio Cabrera
Fecha de creación	21/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema almacena permanentemente (en archivo) los valores establecidos en la lista de configuración de funciones CRUD.
Dependencias	Caso de prueba CPR-1 de prueba PR-2 Caso de prueba CPR-1 de prueba PR-8

Paso	1
Usuario	Usuario acciona evento “save crud” (guardar crud).
Sistema	El sistema almacena lista de configuración y recarga página actual.
Resultado esperado	El sistema almacena permanentemente (en archivo) lista de configuración de funciones CRUD de la totalidad de las tablas del esquema de base de datos. El sistema recarga página actual.

Paso	2
Usuario	Efectue los siguientes casos de prueba (Cerrar sesión) e (Iniciar sesión).
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Seleccione tabla “empleado”.
Sistema	El sistema despliega tabla de configuración de funciones CRUD.
Resultado esperado	El sistema despliega tabla de configuración de funciones CRUD de tabla “empleado”. Los valores de la función CRUD coinciden con los especificados en el caso de prueba CPR-1 de prueba PR-8.

2.2.22. PR-10 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-10
Nombre	Visualizar gráficamente configuración de campos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	21/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema permite la visualización gráfica de las configuraciones establecidas a una tabla del esquema de base de datos.
Dependencias	Caso de prueba CPR-1 de prueba PR-1 Caso de prueba CPR-1 de prueba PR-5

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Resultado esperado	Sistema despliega opción de visualización.

Paso	2
Usuario	Accione evento “Visualizar”.
Sistema	El sistema despliega visualmente las configuraciones.
Resultado esperado	El sistema despliega visualmente las configuraciones de los campos de tabla seleccionada. Las propiedades establecidas en caso de prueba CPR-1 PR-5 coinciden con las visualizadas.

2.2.23. PR-11 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-11
Nombre	Exportar aplicación
Usuario	Héctor Mauricio Cabrera
Fecha de creación	23/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema crea archivos de código fuente con las configuraciones de campos y funciones CRUD establecidas.
Dependencias	Caso de prueba CPR-1 de prueba PR-2 Caso de prueba CPR-1 de prueba PR-6 Caso de prueba CPR-1 de prueba PR-9

Paso	1
Usuario	Seleccione tabla “empleado”.
Resultado esperado	El sistema despliega mecanismo de activación de tabla, configuraciones de campos y funciones CRUD.

Paso	2
Usuario	Active tabla “empleado” accionando mecanismo de activación en parte superior de tabla de configuración de campos.
Resultado esperado	El sistema activa tabla “empleado”.

Paso	3
Usuario	Accione evento “export app” (exportar aplicación).
Sistema	El sistema crea archivos de código fuente para tabla “empleado” activada.
Resultado esperado	El sistema consulta configuraciones y funciones CRUD para tabla “empleado” y genera los archivos de código fuente con el siguiente formato de nombres: nombre_esquema-app.rkt, nombre_esquema-crud.rkt, nombre_esquema-view.rkt. Para este caso son: empven-app.rkt, empven-crud.rkt y empven-view.rkt.

2.2.24. PR-12 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-12
Nombre	Generar mensajes de información
Usuario	Héctor Mauricio Cabrera
Fecha de creación	24/09/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega mensajes de información correctamente.
Dependencias	Caso de prueba CPR-1 de prueba PR-1

Paso	1
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Aquí encontrarás todas las tablas del esquema de base de datos. Selecciona una de ellas para desplegar su tabla de configuración de campos al costado derecho."

Paso	2
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Esta es la tabla de configuración de campos, aquí podrás configurar los campos de cada tabla. Para cambiar la propiedad de algún campo, da click en la celda en que se intersecta la columna de alguna propiedad con la fila de algún nombre de campo. Ten en cuenta que propiedades como tamaño lógico y nulable no son modificables."

Paso	3
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.

Resultado esperado	El sistema despliega mensaje de información: "Aquí puedes modificar el nombre de la etiqueta que acompaña al campo."

Paso	4
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Este será el tipo de elemento HTML que representará al campo. select_date esta diseñado para campos datefecha de la base de datos, lo componen tres elementos select cada uno representando el día, mes y año."

Paso	5
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Aquí puedes establecer el tipo de texto que permitirá el campo. numérico sólo permitirá números [0-9], alfabético sólo permitirá letras del alfabeto [a-zA-Z], alfanumérico sólo letras y números [0-9a-zA-Z], email esta especialmente diseñado para aceptar correos electrónicos y alfanumérico_esp permitirá además caracteres como [@#.,/- espacio]."

Paso	6
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Aquí puedes establecer el tamaño físico del elemento HTML que representa al campo."

Paso	7

Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: “Aquí decides si el campo es necesario mostrarse gráficamente. Será necesario si el campo representa información que el usuario suministra. Por ejemplo: Número de documento, fecha de nacimiento, etc.”

Paso	8
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: “Aquí establecerás las opciones para los campos (select, radio, checkbox) siguiendo el siguiente formato: [“value” “show”],[“v” “s”],...,[“v” “s”] Donde value/v será el valor real del campo y show/s será el valor que se mostrará gráficamente. Las comas que separan a cada opción no deben tener espacios a su alrededor y las comillas dobles siempre serán necesarias.”

Paso	9
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: “Para guardar temporalmente la configuración de los campos, haz click en ’Guardar’. Si deseas guardar permanentemente su configuración, y así en un nuevo ingreso a la aplicación los valores esten conservados, haz click en ’save’ en la parte superior de la herramienta. Ten en cuenta que antes de guardarlos permanentemente debes guardarlos temporalmente.”

Paso	10
Usuario	Seleccione mensaje de información representado con imagen de icono (i).

Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Si deseas visualizar gráficamente la configuración de los campos, haz click en 'Visualizar'."

Paso	11
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Aquí podrás configurar las funciones CRUD (de persistencia) para cada tabla del aplicativo. Para iniciar selecciona el tipo de función que quieres crear. Es necesario crear al menos una función CRUD."

Paso	12
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Aquí establecerás el nombre del procedimiento que dará manejo a la función de persistencia. Este es importante puesto que a través de el se hará la ejecución de la función CRUD. Para nombrar el procedimiento se puede utilizar los caracteres del alfabeto y los simbolos _-/."

Paso	13
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.

Resultado esperado	El sistema despliega mensaje de información: "Aquí podrás seleccionar los campos que intervienen en la función de persistencia. Por ejemplo: Si es la función INSERT, entonces serán los campos que se insertarán. Para UPDATE serán los campos que se actualizarán, DELETE los campos que serán invalidados, y SELECT los campos que serán mostrados."

Paso	14
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Aquí podrás seleccionar los campos que actuarán como restricción en la función de persistencia. Por ejemplo: Si es la función SELECT, serán los campos por los que se hará la búsqueda. Para UPDATE, serán los campos que serán buscados para actualizar. Para DELETE, serán los campos que serán buscados para invalidar. Se recomienda que UPDATE y DELETE tengan restricciones. Si quieres una consulta de todos los registros de una tabla, crea un SELECT sin restricciones."

Paso	15
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Si la restricción a crear involucra más de un campo simultáneamente, debes establecer que conector lógico los une. El conector por defecto es AND."

Paso	16
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.

Resultado esperado	El sistema despliega mensaje de información: "Una vez que tengas la función CRUD especificada podrás guardarla temporalmente a través de 'Guardar', esto creará una tabla en la parte superior que mostrará la función creada. Es importante que una vez hayas creado/editado/eliminado tus funciones CRUD, las guardes de forma permanente a través de 'save crud' en la parte superior de la herramienta, hazlo antes de cambiar a otra tabla o hacer cualquier cosa. Si no lo haces, perderás todo el esfuerzo involucrado en la creación de estas funciones."

Paso	17
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "Aquí podrás activar la tabla si quieres que sea involucrada en la generación del aplicativo. Si no tienes ninguna tabla activa no se generará el aplicativo. Su valor por defecto siempre estará off/apagada cada vez que inicies sesión."

Paso	18
Usuario	Seleccione mensaje de información representado con imagen de icono (i).
Sistema	El sistema despliega mensaje de información relacionado a icono seleccionado.
Resultado esperado	El sistema despliega mensaje de información: "A través de estos lanzadores podrás controlar el almacenamiento de tus configuraciones, generación y ejecución de la aplicación prototipo. Con 'save' almacenarás todas las configuraciones de los campos de la totalidad de las tablas. Con 'save crud' almacenarás todas las funciones CRUD creadas para la tabla actual. Con 'export app' generarás los archivos con el código fuente de la aplicación prototipo incluyendo las configuraciones hechas en los campos y funciones CRUD de las tablas que se encuentran activas y finalmente con 'exec app' pondrás en ejecución la aplicación prototipo."

2.2.25. PR-13 CPR1

Identificador	CPR-1
Identificador de prueba	PR-13
Nombre	Cerrar sesión
Usuario	Héctor Mauricio Cabrera
Fecha de creación	12/09/2015
Descripción	Esta prueba tiene como objetivo verificar el cierre de sesión y abandono del usuario de la herramienta.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Accione el evento "salir" en el menú superior derecho.
Sistema	El sistema elimina los datos de autenticación del usuario (cookies) y abandona la herramienta.
Resultado esperado	El sistema abandona herramienta y despliega página de inicio de sesión.

2.2.26. PR-14 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-14
Nombre	Desplegar tabla dinámica de funciones CRUD personalizadas - Inexistencia de archivo
Usuario	Héctor Mauricio Cabrera
Fecha de creación	24/11/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega correctamente la tabla de configuración de funciones CRUD personalizadas.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Sistema	El sistema no despliega tabla de configuración de funciones CRUD personalizadas.
Resultado esperado	El sistema no despliega tabla de configuración de funciones CRUD personalizadas debido a la no existencia de archivo de configuración.

2.2.27. PR-14 CPR-2

Identificador	CPR-2
Identificador de prueba	PR-14
Nombre	Desplegar tabla dinámica de funciones CRUD personalizadas - Inexistencia de relación
Usuario	Héctor Mauricio Cabrera
Fecha de creación	24/11/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega correctamente la tabla de configuración de funciones CRUD personalizadas.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Sistema	El sistema no despliega tabla de configuración de funciones CRUD personalizadas.
Resultado esperado	El sistema no despliega tabla de configuración de funciones CRUD personalizadas debido a la no existencia de valores relacionados con tabla “empleado”.

2.2.28. PR-14 CPR-3

Identificador	CPR-3
Identificador de prueba	PR-14
Nombre	Desplegar tabla dinámica de funciones CRUD personalizadas - Existencia de archivo
Usuario	Héctor Mauricio Cabrera
Fecha de creación	24/11/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema despliega correctamente la tabla de configuración de funciones CRUD personalizadas.
Dependencias	Caso de prueba CPR-1 de prueba PR-15 Caso de prueba CPR-1 de prueba PR-16

Paso	1
Usuario	Seleccione tabla “empleado” del esquema de base de datos.
Sistema	El sistema despliega tabla de configuración de funciones CRUD personalizadas.
Resultado esperado	El sistema consulta funciones CRUD personalizadas pertenecientes a tabla “empleado” y despliega tabla de configuración.

2.2.29. PR-15 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-15
Nombre	Editar formulario de configuración de funciones CRUD personalizadas - Datos válidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	29/11/2015
Descripción	Esta prueba tiene como objetivo verificar la creación de función CRUD personalizada a una tabla del esquema de base de datos.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla "empleado" y accione el evento 'Función SQL personalizada'.
Resultado esperado	El sistema despliega formulario de configuración de funciones CRUD personalizadas.

Paso	2
Usuario	En el formulario de configuración de funciones CRUD personalizadas especificar los siguientes valores: - Nombre de procedimiento: update-email-empleado - Definición de instrucción SQL: *update* empven.empleado *set* email=$1 *where* numdoc=$2
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Accione el evento "Guardar".
Sistema	El sistema valida autenticidad y registra valores especificados temporalmente.

Resultado esperado	El sistema establece/modifica estructura de datos (lista) en la que almacena la configuración de función CRUD personalizada especificada en paso 1. El sistema despliega tabla de configuración de funciones CRUD personalizadas en la parte superior del formulario.

2.2.30. PR-15 CPR-2

Identificador	CPR-2
Identificador de prueba	PR-15
Nombre	Editar formulario de configuración de funciones CRUD personalizadas - Datos inválidos
Usuario	Héctor Mauricio Cabrera
Fecha de creación	29/11/2015
Descripción	Esta prueba tiene como objetivo validar autenticidad de los datos suministrados en la creación de función CRUD personalizada.
Dependencias	Caso de prueba CPR-1 de prueba PR-2

Paso	1
Usuario	Seleccione tabla “empleado” y accione el evento ’Función SQL personalizada’.
Resultado esperado	El sistema despliega formulario de configuración de funciones CRUD personalizadas.

Paso	2
Usuario	En el formulario de configuración de funciones CRUD personalizadas especificar los siguientes valores: - Nombre de procedimiento: [vacío] - Definición de instrucción SQL: update where
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Accione el evento “Guardar”.
Sistema	El sistema valida autenticidad de datos suministrados.
Resultado esperado	El sistema detiene la creación de función CRUD personalizada y espera a que el usuario especifique campos correctamente o cancele la operación.

Paso	4
Usuario	Accione el evento “Cancelar”.
Sistema	El sistema cancela operación de creación de función CRUD personalizada.
Resultado esperado	El sistema restablece valores del formulario de creación de función CRUD personalizada.

2.2.31. PR-16 CPR-1

Identificador	CPR-1
Identificador de prueba	PR-16
Nombre	Almacenar lista de configuración de funciones CRUD personalizadas
Usuario	Héctor Mauricio Cabrera
Fecha de creación	01/12/2015
Descripción	Esta prueba tiene como objetivo verificar que el sistema almacena permanentemente (en archivo) los valores establecidos en la lista de configuración de funciones CRUD personalizadas.
Dependencias	Caso de prueba CPR-1 de prueba PR-2 Caso de prueba CPR-1 de prueba PR-15

Paso	1
Usuario	Usuario acciona evento “save crud” (guardar crud).
Sistema	El sistema almacena lista de configuración y recarga página actual.
Resultado esperado	El sistema almacena permanentemente (en archivo) lista de configuración de funciones CRUD personalizadas de la totalidad de las tablas del esquema de base de datos. El sistema recarga página actual.

Paso	2
Usuario	Efectue los siguientes casos de prueba (Cerrar sesión) e (Iniciar sesión).
Sistema	N/A
Resultado esperado	N/A

Paso	3
Usuario	Seleccione tabla “empleado”.
Sistema	El sistema despliega tabla de configuración de funciones CRUD personalizadas.
Resultado esperado	El sistema despliega tabla de configuración de funciones CRUD personalizadas de tabla “empleado”. Los valores de la función CRUD coinciden con los especificados en el caso de prueba CPR-1 de prueba PR-15.

2.3. Prueba de usabilidad

Los resultados obtenidos de la encuesta fueron:

1. Una vez revisado los videos y realizado el proceso sugerido, lo que experimentó para generar una aplicación prototipo diría que es:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Muy Rápido	5	1	4.8 %	5
Rápido	4	13	61.9 %	52
Normal	3	4	19 %	12
Lento	2	3	14.3 %	6
Muy lento	1	0	0 %	0
			Calificación promedio	75/21 = 3.57

2. Una vez revisado los videos y realizado el proceso sugerido, diría usted que este es útil para generar una aplicación prototipo:

	Votos	**Porcentaje votos**
Si	21	100 %
No	0	0 %

3. Una vez revisado los videos y seguido el proceso sugerido, diría usted que este es transparente como usuario final, teniendo en cuenta que son macros los que están generando código a partir de un lenguaje de programación en el paradigma funcional.

	Votos	**Porcentaje votos**
Si	20	95.2 %
No	1	4.8 %

4. Los mecanismos que utilizó para generar una aplicación prototipo son:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Muy fáciles de manejar	5	3	14.3 %	15
Fáciles de manejar	3.75	8	38.1 %	30
Normales de manejar	2.50	6	28.6 %	15
Difíciles de manejar	1.25	4	19 %	5
			Calificación promedio	65/21 = 3.09

5. La navegabilidad que utilizó en la herramienta para generar una aplicación prototipo es:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Muy fácil de seguir	5	4	19 %	20
Fácil de seguir	3.75	11	52.4 %	41.25
Normal de seguir	2.50	5	23.8 %	12.5
Difícil de seguir	1.25	1	4.8 %	1.25
			Calificación promedio	75/21 = 3.57

6. Como calificaría la ayuda aportada por la herramienta:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Excelente	5	3	14.3 %	15
Sobresaliente	4	11	52.4 %	44
Buena	3	4	19 %	12
Regular	2	3	14.3 %	6
Mala	1	0	0 %	0
			Calificación promedio	77/21 = 3.66

7. Esta de acuerdo con que el diseño de la interfaz gráfica es:

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Excelente	5	0	0 %	0
Sobresaliente	4	7	33.3 %	28
Bueno	3	7	33.3 %	21
Aceptable	2	5	23.8 %	10
Insuficiente	1	2	9.5 %	2
			Calificación promedio	61/21 = 2.90

8. Qué proceso o procesos se le dificultó más en entender y aplicar:

	Votos	**Porcentaje votos**
Configuración de campos	3	9 %
Configuración de funciones CRUD	14	41 %
Almacenamiento de configuración de campos	5	15 %
Almacenamiento de funciones CRUD	7	20 %
Visualización de configuración de campos	1	3 %
Exportar aplicación	1	3 %
Ejecutar aplicación	3	9 %

9. Está de acuerdo con el mecanismo de edición de la tabla de configuración de campos.

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Completamente de acuerdo	5	5	23.8 %	25
En mayor parte de acuerdo	4	8	38.1 %	32
En buena parte de acuerdo	3	4	19 %	12
Poco de acuerdo	2	3	14.3 %	6
Nada de acuerdo	1	1	4.8 %	1
			Calificación promedio	76/21 = 3.61

10. Está de acuerdo con el mecanismo de edición de funciones CRUD (crear, leer, actualizar y eliminar).

	Calificación	**Votos**	**Porcentaje votos**	**Calificación total**
Completamente de acuerdo	5	7	33.3 %	35
En mayor parte de acuerdo	4	6	28.6 %	24
En buena parte de acuerdo	3	6	28.6 %	18
Poco de acuerdo	2	2	9.5 %	4
Nada de acuerdo	1	0	0 %	0
			Calificación promedio	81/21 = 3.85

11. Ha tenido experiencia con alguna aplicación similar a la herramienta que aquí experimentó.

	Votos	**Porcentaje votos**
Si	8	38.1 %
No	13	61.9 %

12. Justifica el uso de la herramienta para obtener los resultados que ella genera.

	Votos	**Porcentaje votos**
Si	19	90.5 %
No	2	9.5 %

13. Califique en porcentaje entre 0 % y 100 % su grado de satisfacción en términos generales, sobre el uso de la aplicación, siendo 0 % nada satisfecho y 100 % completamente satisfecho.

Rango de satisfacción	**Votos**	**Porcentaje votos**	**Total grado satisfacción**
0	0	0 %	0
10	0	0 %	0
20	0	0 %	0
30	0	0 %	0
40	2	9.5 %	80
50	1	4.8 %	50
60	1	4.8 %	60
70	3	14.3 %	210
80	8	38.1 %	640
90	6	28.6 %	540
100	0	0 %	0
		Promedio grado satisfacción	1580/21 = 75.23

14. Presentó algún problema en el uso de la herramienta, si fue así podría especificarlo.

- Ninguno.
- No presenté ningún problema con la herramienta.
- No.
- Para un usuario final que no tenga ningún tipo de experiencia con este tipo de aplicaciones sería ideal colocar terminología común o general. ej. "función de persistencia" podría cambiar a "Almacenamiento".
- En un momento cuando estaba en la aplicación(primeros pasos) al dar click en guardar no realizó la operación respectiva. -por lo que en efecto tocó volver a comenzar-.
- Problema con la implementación del CRUD.
- Ninguno.
- Ninguno.
- No, no se presentó problema.
- Ninguno.
- Al momento de editar la propiedad "mostrar" el campo numdoc de la tabla empleados noté que no se guardaban los cambios entre visible y no visible, para esto se requirió hacer uso de las opciones save y guardar una precedida de la otra. De esta manera reconoció los cambios.
- Ninguno.
- No pude ingresar las opciones del iddpto y idciudad.
- No pude generar la aplicación, seguí los pasos en los vídeos pero nunca me cargó la aplicación generada, incluso después de varios minutos.
- Al momento de seleccionar la opción "save crud", las crud sencillas creadas se eliminaban de la lista, solo almacenaba las crud personalizadas. Se elimina caché y se cambia de navegador corrigiendo así el problema.
- No.

- En la lista de opciones, van enunciados por [] y se tiene que separar por comas sin dejar espacio, de lo contrario no funciona, lo cual no es para nada intuitivo, ademas no muestra advertencias o errores. Luego de dar visualizar, el botón regresar no devuelve a la página que se estaba editando. Si a una opción le selecciono "radio" pero no le configuro las opciones lo toma por defecto como "text" y no obliga a la configuración.
- No.
- Ninguno.
- Al registrarme, fallé varias veces al crear la cuenta, por la cantidad de nombres y apellidos que ingresaba.

15. Como valoramos su experiencia con la herramienta, y queremos que esta sea lo mejor posible, por favor déjenos su comentario, crítica constructiva y/o noción de cambio.

- Tal vez mejoraría la parte de diseño de la herramienta. Pero a nivel funcional me parece que es muy completa y muy útil.
- Implementación de cabecera en la tabla de exportación.
- Aunque no es una camisa de fuerza para poder generar un prototipo si es muy importante el diseño, tanto para la configuración de campos y funciones crud como para la visualización de los mismos.
- Si los vídeos tuvieran audio sería mucho más fácil de entender el propósito de la aplicación.
- Existen herramientas mbd mucho mas prácticas y más visuales para el usuario final, siendo esta una tendencia en programación.
- Mejorar los mensajes dentro de la aplicación.
- A nivel general, si el usuario es desarrollador se entiende la lógica de lo que se debe hacer, pero para alguien que no programe va a tener dificultades. Hay una gran oportunidad de mejoría en cuanto a la interfaz de usuario y su usabilidad, debería ser transparente por ejemplo en opciones tener que hacer cosas como '[“llave” “valor”],[“” “”],..'. Pueden mejorar las ayudas siendo un poco más descriptivas y en lo posible mostrar ejemplos. No es claro que se tenga que dar click en los campos para modificar su configuración.
- En el proceso de configuración de campos se podría agilizar guardando automáticamente cada configuración o (pulsando enter), la especificación de la lista de opciones no es agradable y se congestiona mucho por las comas, comillas y corchetes. Estoy de acuerdo con la configuración de funciones CRUD personalizada en lugar de la normal. El proceso de almacenamiento para ambos procesos debería ser en un solo paso. Finalmente las ayudas son poco atractivas y se dificulta su lectura.
- Se puede mejorar la usabilidad de la herramienta, en la interfaz de configuración de campos hay tres opciones que parecen iguales: “Save”, “Save CRUD” y “Guardar”. Sería muy útil tener un mensaje que avise cuando la aplicación ya está disponible porque al presionar “exec app” la página se recarga y uno no sabe que pasa. Tal vez las acciones “export app” y “exec app” podrían ser combinados en una sola opción.

- Cuando se están ingresando los datos de la tabla y se comete algún error, se debe realizar una descripción mejor del error que se está cometiendo. - La descripción de los errores cometidos al llenar el formulario deberían desaparacer al momento de ser corregidos. - En la creación de las crud debería haber una opción de select all, en caso de que hayan muchos campos que seleccionar.

- La configuración de campos y las funciones CRUD se deberían generar automáticamente a partir del modelo de la base de datos. Es interesante la opción de configurar funciones CRUD personalizadas, pero las básicas se deberían generar a partir de la base de datos. Otro punto que a mi parecer esta muy básico, es la interfaz gráfica final, muy pobre en cuanto a diseño.

- Buena herramienta.

- Se podría validar para la configuración de los campos la manera de configurar las opciones de los mismos. por ejemplo, no tener que dar doble click para que las opciones aparezcan, si no que ya estén disponibles para hacer los cambios requeridos. Para crear un SQL personalizado, se puede crear la opción de seleccionar directamente los campos que irían en el SELECT, o en el UPDATE o DELETE, al igual que en el WHERE sin necesidad de escribir el código SQL.

- La herramienta me parece útil en la recolección de datos, por su rápida y fácil organización de los mismos, Los vídeos instructivos son de bastante ayuda, aunque en esta situación reina lo visual (en principio pasé por alto los comentarios hechos sobre el vídeo y creía necesario una parte auditiva).

- Está limitado los campos correo, no se está usando UTF-8, faltan avisos de error al ingresar los datos, si son erroneos. Faltan conectores lógicos como xor, like, not, as en la parte de crear las funciones CRUD, me pareció genial lo que se logró con racket.

- Sería muy importante el poder tener la herramienta sin los bloqueos de herramienta en uso.

- Es una herramienta con mucho potencial que se podría desarrollar más con el fin de que ofrezca muchas mas opciones. Inicialmente pienso que es muy buena y podría mejorarse de alguna forma la usabilidad, para que personas no ingenieros de sistemas o afines pudieran usarla. LDP

- Solo que los colores y estilo de letra de la GUI sean mejores, actualmente se ven poco llamativos.

- El ser monousuario limita su uso en la práctica.

- Creo que se debe mejorar la navegabilidad durante la configuración del crud, que el panel de configuraciones personalizadas esté separado del panel de creación inicial de funciones crud.

Capítulo 3

Documentación de funciones

3.1. Funciones en módulo campos

campos
inic-campos :: string → (listof list?)
(inic-campos schema)
Consulta las propiedades de los campos de cada tabla del esquema, si exite archivo con esas configuraciones, las carga desde este, de lo contrario lo hace desde la base de datos. Retorna lista de listas.

campos
lstvec-to-lst :: (listof vector?) → (listof list?)
(lstvec-to-lst lstvec)
Convierte lista de vectores en lista de listas.

campos
save-all-lst :: string (listof list?) → void?
(save-all-lst table lst)
Si estructura vcamposg se encuentra vacío, se agrega lista de listas de propiedades de los campos de tabla especificada, de lo contrario si vcamposg contiene datos, pero no las propiedades de los campos de la tabla especificada, de nuevo se agregan, en caso contrario se advierte que las propiedades de los campos de esa tabla ya existen. Retorna void?.

campos
add-elemts :: (listof list?) → (listof list?)
(add-elemts lst)
Construye lista de listas de las propiedades de los campos de cada tabla, sobre estructura vcamposg. Retorna lista de listas.

campos
exist-lst :: string (listof list?) → boolean
(exist-lst table lst)
Consulta si existen propiedades de los campos de tabla especificada en estructura vcamposg. Si es así, retorna #t, de lo contrario #f.

campos
lst-conf :: (listof list?) → (listof list?)
(lst-conf lsta)
Se toma lista de listas de propiedades de los campos de cada tabla, que fue generada previamente desde la base de datos y se complementan con otras propiedades. Retorna lista de listas.

campos
get-tipo-elemt :: string → string
(get-tipo-elemt elemt)
Renombra tipos de datos de la base de datos a tipos de datos utilizados en la herramienta. Retorna string.

campos
get-tam-elemt :: string number → string
(get-tam-elemt tipoelemt tam)
Establece tamaño adecuado para tipo de dato select_date, y tamaño por defecto para los demás elementos en caso de no estar establecido. Retorna string.

campos
save-schema-file :: string → void?
(save-schema-file schema)
Almacena en archivo schema.data los valores contenidos en estructura vcamposg. Retorna void?.

campos
get-schema-file :: string → boolean
(get-schema-file schema)
Si archivo con nombre schema.data existe, carga estructura vcamposg con los datos almacenados en este. Retorna #t si fue así, #f si no existe archivo.

campos
load-tables :: string → list?
(load-tables schema)
Consulta campos y propiedades por cada tabla perteneciente al esquema, la lista de vectores obtenida se convierte en lista de listas, se agregan nuevas propiedades formando la lista de configuración de campos, que se almacena en estructura vcamposg. Retorna lista.

campos
load-data-tables :: string → (listof list?)
(load-data-tables schema)
Consulta campos y propiedades por cada tabla perteneciente al esquema, y crea estructura adecuada que almacena en vcamposg. Retorna lista de listas.

campos
get-vcampos :: string string → (listof list?)
(get-vcampos schema table)
Obtiene lista de listas de propiedades de los campos de tabla especificada desde estructura vcamposg. Retorna lista de listas.

campos
get-lst :: string (listof list?) → (listof list?)
(get-lst table lst)
Construye lista de listas de propiedades de los campos de tabla especificada desde estructura vcamposg. Retorna lista de listas.

campos
prepare-new-lst :: string string list? → (listof list?)
(prepare-new-lst schema table str)

Toma expresión en cadena de caracteres proveniente de la tabla de configuración de campos editada por el usuario (formateada en javascript) y la transforma en lista de listas. Retorna lista de listas.

campos
fix-fields :: list? → (listof list?)
(fix-fields lst)
Construye lista de listas de las propiedades de los campos de una tabla del esquema, con la información proveniente de la tabla de configuración de campos editada por el usuario (formateada en javascript). Retorna lista de listas.

campos
real-opcs :: string → (listof list?)
(real-opcs str)
Construye lista de listas de valores para las opciones que deben especificarse en los elementos select, checkbox o radio. Retorna lista de listas.

campos
modf-lst :: string string (listof list?) → void?
(modf-lst schema table lst)
Modifica las propiedades de los campos de tabla especificada sobre estructura vcamposg. Retorna void?.

campos
clean-lst :: string (listof list?) → (listof list?)
(clena-lst table lst)
Elimina las propiedades de los campos de tabla especificada sobre estructura vcamposg. Retorna lista de listas.

campos
flatten-lst :: (listof list?) → list?
(flatten-lst lst)
Encargado de aplanar una lista de listas a un nivel más bajo. Retorna lista.

campos
get-campos-td-aux :: string → (listof list?)

(get-campos-td-aux schema)
Construye una lista de listas con los tipos de datos de los campos de cada una de las tablas del esquema especificado. Retorna una lista de listas.

campos
get-campo-td :: string string string → list?
(get-campo-td schema ntable field)
Obtiene el tipo de dato de un campo para un esquema, tabla y nombre de campo especificado. Retorna lista.

campos
render-vcampos-table-conf :: (listof list?) number → list?
(render-vcampos-table-conf lsta n)
Extrae cada valor de las propiedades de los campos de una tabla del esquema y forma lista de elementos html <tr>, que forman la tabla de configuración de campos. Retorna lista.

campos
render-opcs :: (listof list?) → string
(render-opcs opcs)
Convierte lista de los valores de las opciones en cadena de texto, para luego ser desplegados en la tabla de configuración de campos. Retorna string.

campos
clean-campos-data :: nothing → void?
(clean-campos-data)
Elimina datos almacenados en estructura vcamposg. Retorna void?.

3.2. Funciones en módulo codegen

codegen
start :: request? → expr?/boolean
(start request)
Inicia conexión con base de datos y dirige la ejecución de la herramienta a la página de inicio. Retorna expresión. En caso de no ser posible iniciar la base de datos retorna #f.

codegen
codegen-dispatch :: url → expr?
(codegen-dispatch codegen-url)
Selecciona regla de despacho según página web solicitada por el usuario, que pertenece a la herramienta CASE. Retorna expresión.

codegen
codegen-login-tmp :: request → expr?
(codegen-login-tmp request)
Especificación de regla de despacho para página denominada login en herramienta CASE. Retorna expresión.

codegen
codegen-register-tmp :: request → expr?
(codegen-register-tmp request)
Especificación de regla de despacho para página denominada register en herramienta CASE. Retorna expresión.

codegen
codegen-register :: string request → expr?
(codegen-register args request)
Despliega página HTML de registro para la creación de un nuevo usuario. Retorna expresión.

codegen
make-reg :: request → expr?
(make-reg request)

Captura información de registro especificada por el usuario e inicia proceso de registro. Retorna expresión.

codegen
codegen-login :: string request → expr?
(codegen-login args request)
Despliega página HTML para la autenticación e ingreso del usuario a la herramienta CASE. Retorna expresión.

codegen
make-login :: request → expr?
(make-login request)
Captura datos especificados por el usuario necesarios para iniciar sesión, comprueba su autenticidad ante la base de datos, y de ser correcto crea cookies para la sesión y redirecciona al usuario a la página principal. De lo contrario no inicia sesión y advierte al usuario a través de un mensaje de error. Retorna expresión.

codegen
make-options :: string → expr?
(make-options elemt)
Crea elemento HTML option. Retorna expresión.

codegen
make-select :: list? → expr?
(make-select lst)
Crea elemento HTML select con los valores de la lista como opciones. Retorna expresión.

codegen
codegen-index :: string string string request → expr?
(codegen-index id schema ntable request)
Despliega página principal de la herramienta CASE. Lista tablas del esquema, crea tabla de configuración de campos, tabla de funciones CRUD y formularios de configuración de funciones CRUD. Retorna expresión.

codegen

make-table :: request → expr?
(make-table request)
Carga página principal con tabla del esquema de base de datos seleccionada por el usuario. Retorna expresión.

codegen
modf-table :: request → expr?
(modf-table request)
Captura nombre de tabla, configuración de campos y modifica estructura de datos para almacenar la nueva información. Retorna expresión.

codegen
save-schema :: request → expr?
(save-schema request)
Almacena en archivo las configuraciones de los campos de la totalidad de las tablas. Retorna expresión.

codegen
save-crud :: request → expr?
(save-crud request)
Captura configuración de función CRUD especificada, modifica estructura de datos y almacena en archivo funciones CRUD de la totalidad de las tablas incluyendo la modificación previa. Retorna expresión.

codegen
save-crud-per :: request → expr?
(save-crud-per request)
Captura configuración de función CRUD personalizada, modifica estructura de datos y almacena en archivo la totalidad de las funciones CRUD personalizadas, incluyendo la modificación previa. Retorna expresión.

codegen
export-app :: request → expr?
(export-app request)
Genera archivos de código fuente de aplicación prototipo, en base a las configuraciones de campos y funciones CRUD especificadas. Retorna expresión.

codegen
exec-app :: request → expr?
(exec-app request)
Inicia servidor web y aplicación web prototipo previamente generada, a la espera de peticiones del usuario. Retorna expresión.

codegen
set-table :: request → expr?
(set-table request)
Activa o desactiva tabla del esquema de base datos. De acuerdo a su estado, se genera código fuente asociado. Retorna expresión.

codegen
view-table :: request → expr?
(view-table request)
Despliega visualmente las configuraciones de campos hecha en la tabla del esquema actualmente seleccionada. Retorna expresión.

codegen
logout :: request → expr?
(logout request)
Cierra sesión del usuario, elimina cookies relacionadas. Retorna expresión.

codegen
match-remove-crud :: string → boolean
(match-remove-crud str)
Verifica si cadena de texto entrante contiene el valor "remove_crud_table", para dar inicio a la eliminación de las configuraciones de funciones CRUD, de una tabla en especifico. Retorna #t si es así, #f en caso contrario.

codegen
match-remove-crud-per :: string → boolean
(match-remove-crud-per str)
Verifica si cadena de texto entrante contiene el valor "remove_crud_per_table", para dar inicio a la eliminación de las configuraciones de funciones CRUD personalizadas, de una tabla en especifico. Retorna #t si es así, #f en caso contrario.

codegen
kill :: string → void?
(kill app)
Detiene ejecución de aplicación web prototipo previamente iniciada con procedimiento yes-exec-app. Retorna #t si instrucción a través de system se ejecutó con éxito, #f en caso contrario.

codegen
yes-exec-app :: string → list?
(yes-exec-app sch)
Inicia ejecución de aplicación web prototipo, en caso de haber sido ya iniciada, cierra e inicia un nuevo proceso de ejecución. Retorna una lista.

codegen
yes-kill-app :: string → boolean
(yes-kill-app sch)
Inicia la detención de aplicación web prototipo previamente inicializada. Retorna #t si se detuvo la aplicación, #f en caso contrario.

codegen
render-tablas :: string string string list? → expr?
(render-tablas idt str1 str2 lst)
Despliega gráficamente las tablas pertenecientes al esquema de base de datos. Retona expresión. Basa su despliegue en el elemento HTML ul.

codegen
elemt-ul :: string list? → expr?
(elemt-ul idt lst)
Crea elemento HTML ul con subelementos li para los valores de la lista. Retorna expresión.

codegen
elemt-li :: string → expr?
(elemt-li elemt)
Crea elemento HTML li para cada valor especificado. Retorna expresión.

codegen
render-crud-index :: string string (listof list?) → expr?
(render-crud-index schema ntable vcampos)
Despliega formulario de configuración de funciones CRUD. Retorna expresión.

codegen
render-vcampos-list :: (listof list?) → list?
(render-vcampos-list lsta)
Crea elemento HTML checkbox para cada uno de los campos de la tabla del esquema de base de datos y los enlista. Retorna lista.

codegen
render-crud-per-index :: string string (listof list?) → expr?
(render-crud-per-index schema ntable vcampos)
Despliega formulario de configuración de funciones CRUD personalizadas. Retorna expresión.

codegen
render-vcampos-listtext :: string (listof list?) → string
(render-vcampos-listtext str lsta)
Convierte lista de elementos en una cadena de texto. Retorna cadena de texto.

codegen
codegen-vista :: string string request → expr?
(codegen-vista id schema ntable)
Despliega página que visualiza la configuración de los campos de la tabla seleccionada. Retorna expresión.

codegen
sub-str :: string → string
(sub-str str)
Elimina últimos 3 caracteres =\r\n de cadena especificada. Retorna string.

codegen
make-digest :: string string → string
(make-digest str1 str2)

Codifica valor de las dos cadenas de texto. Retorna string.

codegen
del-auth-cookie :: string string string → list?
(del-auth-cookie user schema table)
Expira y elimina cookies que soportan la sesión del usuario. Retorna lista.

codegen
make-auth-cookie :: string string string → list?
(make-auth-cookie user schema table)
Crea cookies para identificación del usuario a través de la herramienta. Retorna string.

codegen
extract-auth-cookies :: request → list?
(extract-auth-cookies req)
Recupera y decodifica cookies para autenticación del usuario. Si es correcto, se retorna lista con el contenido de las cookies, de lo contrario se retorna #f.

codegen
make-auth-cookie-then-redirect-to :: string string string → expr?
(make-auth-cookie-then-redirect-to user schema table where-to)
Inicia creación de cookies para identificación de usuario y redirige a página indicada. Retorna expresión.

codegen
del-auth-cookie-then-redirect-to :: string string string → expr?
(del-auth-cookie-then-redirect-to user schema table where-to)
Elimina cookies para cierre de sesión y lograr salida del usuario de la herramienta. Retorna expresión.

codegen
auth :: request → list?/boolean
(auth req)
Comprueba autenticidad del usuario a través de las cookies, de ser correcto se retorna lista con sus valores, de lo contrario se retorna #f.

3.3. Funciones en módulo conn

conn
inicializar-conn :: nothing → boolean
(inicializar-conn)
Establece conexion con base de datos, de ser así devuelve #t, de lo contrario #f.

conn
inicializado :: nothing → boolean
(inicializado)
Verifica si existe conexión establecida con base de datos, si es así devuelve #t, de lo contrario #f.

conn
manage-error-sql :: list? → string
(manage-error-sql lst)
Procesa errores originados en las funciones de persistencia. Retorna string.

conn
insert-new-user :: string string string string string string → string/boolean
(insert-new-user email nom ape sexo pass confpass)
Registra nuevo cliente en la base de datos. Si se crea un nuevo usuario se retorna cadena de texto confirmando operación, Si ocurre error en la operación se retorna cadena de texto que lo indica, de lo contrario #f.

conn
insert-relation :: string string string → string/boolean
(insert-relation id esqa esqb)
Relaciona un nuevo usuario con dos esquemas de base de datos pre-establecidos. Retorna cadena de texto en caso de ejecutar la operación, #f en caso de no hacerlo.

conn
login-check-schema-inbd :: string → string/boolean
(login-check-schema-inbd schema)
Consulta si esquema está creado en la base de datos, no como registro en tabla esquemas. Si existe el esquema, se retorna su nombre de lo contrario #f.

conn
select-schema :: nothing → list?/boolean
(select-schema)
Consulta todos los esquemas registrados en tabla esquema. Retorna lista si operación es exitosa, de lo contrario #f.

conn
login-check-user :: string string → boolean
(login-check-user user pass)
Consulta si usuario se encuentra registrado. Retorna #t de ser así, #f en caso contrario.

conn
login-check-schema :: string string → boolean
(login-check-schema user schema)
Consulta si esquema se encuentra asociado al usuario. Retorna #t de ser así, #f en caso contrario.

conn
select-user :: string string → list?
(select-user user pass)
Consulta información de usuario. Retorna lista si existe, #f en caso contrario.

conn
get-tablas :: string → list?
(get-tablas schema)
Consulta tablas pertenecientes a un esquema, retorna lista con los nombres de las tablas.

conn
get-campos :: string string → (listof vector?)
(get-campos schema table)
Consulta campos y propiedades de una tabla perteneciente a un esquema, retorna una lista de vectores con los campos y propiedades.

conn

get-tipodatos-campos :: string string → (listof vector?)
(get-tipodatos-campos schema table)
Consulta los tipos de datos de los campos de una tabla y esquema en especifico. Retorna una lista de vectores.

conn
login-app :: nothing → boolean
(login-app)
Consulta estado actual de la herramienta, si está en uso retorna #t, #f en caso contrario.

conn
set-login-app :: boolean
(set-login-app value)
Cambia estado actual de la herramienta, #t si empezó a ser usada, #f en caso contrario.

3.4. Funciones en módulo crud-format

crud-format
lst-split :: (listof string) → (listof list?)
(lst-split lst)
Convierte funciones CRUD de cada tabla del esquema especificados por el usuario (Cadena de texto formateada en javascript), en lista de listas de funciones CRUD. Retorna lista de listas.

crud-format
lst-split-esp :: list? → (listof list?)
(lst-split-esp lst)
Convierte funciones CRUD personalizadas de cada tabla del esquema especificado por el usuario (Cadena de texto formateada en javascript), en lista de listas de funciones CRUD personalizadas. Retorna lista de listas.

crud-format
format-sql :: (listof list?) → (listof list?)
(format-sql lst)
Construye formato adecuado para cada instrucción SQL (función CRUD) de cada tabla del esquema, que ha sido especificada. Retorna lista de listas.

crud-format
format-sql-norest :: list? → list?
(format-sql-norest lst)
Construye instrucciones INSERT y SELECT sin restricciones. Retorna lista.

crud-format
format-insert :: string string string → (listof string list?)
(format-insert schema ntable campos)
Construye instrucción INSERT con campos especificados, para una tabla del esquema. Retorna lista con elemento string y lista.

crud-format
format-select :: string string string → string
(format-select schema ntable campos)

Construye instrucción SELECT sin restricciones, con campos especificados, para una tabla del esquema. Retorna string.

crud-format
format-update :: string string string → string
(format-update schema ntable campos)
Construye instrucción UPDATE sin restricciones, con campos especificados, para una tabla del esquema. Retorna string.

crud-format
format-sql-rest :: list? → list?
(format-sql-rest lst)
Construye instrucciones SELECT, UPDATE o DELETE con restricciones. Retorna lista.

crud-format
format-select-rest :: string string string string → (listof string list?)
(format-select-rest schema ntable campos rests)
Construye instrucción SELECT con restricciones, con campos especificados, para una tabla del esquema. Retorna lista con elemento string y lista.

crud-format
format-update-rest :: string string string string → (listof string list?)
(format-update-rest schema ntable campos rests)
Construye instrucción UPDATE con restricciones, con campos especificados, para una tabla del esquema. Retorna lista con elemento string y lista.

crud-format
format-delete-rest :: string string string string → (listof string list?)
(format-delete-rest schema ntable campos rests)
Construye instrucción DELETE a través de instrucción UPDATE con restricciones, campos especificados, para una tabla del esquema. Retorna lista con elemento string y lista.

crud-format
format-rest :: string string string → string

(format-rest schema ntable rests)
Construye restricciones para instrucción SQL que lo necesite. Retorna string.

crud-format
format-rest-aux :: list? → string
(format-rest-aux lst)
Convierte lista de restricciones a cadena de texto. Retorna string.

crud-format
format-rest-many :: list? → string
(format-rest-many lst)
Convierte lista compuesta de restricciones en cadena de texto. Retorna string.

crud-format
format-rest-one :: list? → string
(format-rest-one lst)
Convierte lista simple de restricciones en cadena de texto. Retorna string.

crud-format
fmt-many :: string string list? → string
(fmt-many s opr lst)
Construye cadena de texto de restricciones a partir de una lista compuesta de restricciones. Retorna string.

crud-format
get-markers :: number → string
(get-markers long)
Construye cadena de texto de marcadores $ con la cantidad especificada. Retorna string.

crud-format
get-update-markers :: string list? → string
(get-update-markers s lst)
Construye cadena de texto con marcadores ? para cada elemento de la lista. Retorna string.

crud-format
get-lst-sym :: list? → list?
(get-lst-sym lst)
Obtiene lista de campos o restricciones. Retorna lista.

crud-format
to-sym :: list? → (listof symbol?)
(to-sym lst)
Convierte lista de elementos string a lista de simbolos. Retorna lista de simbolos.

crud-format
fix-markers-aux :: number string → string
(fix-markers-aux n str)
Convierte marcador ? a $# según la posición que ubique. Retorna string.

crud-format
fix-markers :: string → string
(fix-markers str)
Inicia conversión de marcadores ? en $#. Retorna string.

crud-format
fix-del :: list? → list?
(fix-del lst)
Adiciona a cada valor de la lista la cadena _del. Retorna lista.

3.5. Funciones en módulo crud-funcs

crud-funcs
save-crud-funcs-file :: string list? → void?
(save-crud-funcs-file schema lst)
Crea archivo de código fuente schema-crud.rkt con funciones de persistencia para el aplicativo prototipo, para las tablas del esquema que fueron especificadas. Retorna void?.

crud-funcs
remove-crud-funcs-file :: string → void?
(remove-crud-funcs-file schema)
Elimina archivo de código fuente de funciones de persistencia schema-crud.rkt. Retorna void?.

crud-funcs
write-crud-funcs-file :: open-output-file list? → void?
(write-crud-funcs-file df lst)
Escribe funciones de persistencia en archivo schema-crud.rkt. Retorna void?.

crud-funcs
func-conn :: nothing → expr?
(func-conn)
Crea expresión para conexión con base de datos del aplicativo generado. Retorna expresión.

crud-funcs
check-conn :: nothing → expr?
(check-conn)
Crea expresión para verificación de conexión con base de datos del aplicativo generado. Retorna expresión.

crud-funcs
error-sql :: nothing → expr?
(error-sql)
Crea expresión para manejo de errores de las funciones de persistencia del aplicativo generado. Retorna expresión.

crud-funcs
write-head :: open-output-file string → void?
(write-head df schema)
Crea y escribe funciones cabecera en archivo de persistencia del aplicativo generado. Retorna void?.

crud-funcs
write-end :: open-output-file → void?
(write-end df)
Escribe función de provisión para las funciones definidas en el archivo de persistencia. Retorna void?.

crud-funcs
format-date :: string → expr?
(format-date)
Crea expresión para la conversión de valores date (fecha) en string a estructura make-sql-date. Retorna expresión.

crud-funcs
find-term :: string → boolean
(find-term mydate)
Busca si campo especificado corresponde al campo date (fecha) en la base de datos. Retorna #t si es así, #f en caso contrario.

crud-funcs
format-args :: list? → list?
(format-args args)
Busca si en lista recibida existe un campo date (fecha) para especificar función de conversión. Retorna lista.

crud-funcs
format-args-td :: string string list? → expr?
(format-args-td sch tab args)
Extrae el tipo de datos de cada campo de la lista otorgada, para especificar la expresión de conversión adecuada. Retorna expresión.

crud-funcs
make-crud-funcs :: (listof list?) → list?
(make-crud-funcs lst)
Crea expresiones para las funciones CRUD especificadas de las tablas del esquema. Retorna lista.

crud-funcs
make-crud-func :: list? → expr?
(make-crud-func elemt)
Obtiene el tamaño de la lista especificada, para decidir que tipo de función CRUD crear (Con o sin restricciones). Retorna expresión.

crud-funcs
make-crud-func-nargs :: string string string → expr?
(make-crud-func-nargs nfunc nsql func)
Decide que tipo de función CRUD sin restricciones debe crear. Retorna expresión.

crud-funcs
make-crud-func-args :: string string list? → expr?
(make-crud-func-args nfunc nsql func args)
Decide que tipo de función CRUD con restricciones debe crear. Retorna expresión.

crud-funcs
sql-select-nargs :: string string → expr?
(sql-select-nargs nfunc func)
Crea expresión de función SELECT sin argumentos, que posteriormente se escribe en archivo de persistencia del aplicativo generado. Retorna expresión.

crud-funcs
sql-insert-args :: string string list? → expr?
(sql-insert-args nfunc func args)
Crea expresión de función INSERT con argumentos, que posteriormente se escribe en archivo de persistencia del aplicativo generado. Retorna expresión.

crud-funcs
sql-select-args :: string string list? → expr?

(sql-select-args nfunc func args)
Crea expresión de función SELECT con argumentos, que posteriormente se escribe en archivo de persistencia del aplicativo generado. Retorna expresión.

crud-funcs
sql-update-args :: string string list? → expr?
(sql-update-args nfunc func args)
Crea expresión de función UPDATE con argumentos, que posteriormente se escribe en archivo de persistencia del aplicativo generado. Retorna expresión.

3.6. Funciones en módulo crud-render

crud-render
block :: list? → list?
(block lst)
Especifica asignación de valores a variables, que son tomados de formulario HTML relacionado a una función CRUD. Retorna lista.

crud-render
make-block :: string string list? → expr?
(make-block npg func lst)
Crea asignación de variables y llamado a función CRUD de una tabla del esquema en especifico. Retorna expresión.

crud-render
lst-for-block :: string string list? → (listof list?)
(lst-for-block npg ntb lst)
Construye lista de listas de valores de asignación y llamado de funciones CRUD de una tabla en especifico, para posteriormente crear sus expresiones. Retorna lista de listas.

crud-render
prev-make-block :: string string (listof list?) → list?
(prev-make-block npg ntb lst)
Busca y crea expresiones de asignación y llamado de funciones CRUD de una tabla en especifico. Retorna lista.

crud-render
lst-for-block-form :: string string list? → (listof list?)
(lst-for-block-form npg ntb lst)
Construye lista de listas con nombres de campos y nombre de función CRUD de una tabla en especifico, para posteriormente crear expresión del formulario. Retorna lista de listas.

crud-render
block-form :: string string list? → list?
(block-form func jsform lst)

Crea elementos HTML input para formulario de función CRUD perteneciente a una tabla en especifico. Retorna lista.

crud-render
make-block-form :: string string string list? → expr?
(make-block-form npg func typef lst)
Construye formulario de una función CRUD de una tabla en especifico. Retorna expresión.

crud-render
prev-make-block-form :: string string list? → list?
(prev-make-block-form npg ntb lst)
Busca y crea formularios para todas las funciones CRUD de una tabla en especifico. Retorna lista.

crud-render
lst-for-block-sa :: string string list? → list?
(lst-for-block-sa npg ntb lst)
Crea lista con valores de asignación de funcion SELECT sin restricción de una tabla en especifico. Retorna lista.

crud-render
make-block-sa :: string → list?
(make-block-sa nfunc)
Crea expresión de llamado a función SELECT sin restricciones. Retorna lista.

crud-render
prev-make-block-no-form :: string string list? → list?
(prev-make-block-no-form npg ntb lst)
Busca y crea expresiones de asignación y llamado de función SELECT sin restricciones de una tabla en especifico. Retorna lista.

3.7. Funciones en módulo crud

crud
save-crud-file :: string string string → void?
(save-crud-file schema extfile data)
Almacena en archivo las configuraciones de funciones CRUD estándar y personalizadas de la totalidad de las tablas del esquema, haciendo distinción del archivo a través de su extensión. Retorna void?.

crud
get-crud-file :: string → void?
(get-crud-file schema)
Obtiene configuraciones de funciones CRUD desde archivo schema.crud, y los asigna a estructura vcrudg. Retorna void?.

crud
get-crud-per-file :: string → void?
(get-crud-per-file schema)
Obtiene configuraciones de funciones CRUD personalizadas desde archivo schema.crudp, y los asigna a estructura vcrudg-per. Retorna void?.

crud
remove-crud-file :: string string → void?
(remove-crud-file schema extfile)
Elimina archivo de configuración de funciones CRUD estándar y personalizadas haciendo distinción del archivo por su extensión. Retorna void?.

crud
save-crud-file-a :: string string → void?
(save-crud-file-a schema type)
Inicia proceso de almacenamiento de las funciones CRUD estándar y personalizadas. Retorna void?

crud
prepare-data-crud :: string string string → (listof list?)
(prepare-data-crud schema ntable datacrud)

Toma expresión en cadena de caracteres proveniente del formulario de configuración de funciones CRUD creada por el usuario (formateada en javascript) y la transforma en lista de listas. Retorna lista de listas.

crud
clean-crud :: string (listof list?) → (listof list?)
(clean-crud ntable lst)
Elimina configuraciones de funciones CRUD de una tabla en especifico. Retorna lista de listas.

crud
add-crud :: (listof list?) → (listof list?)
(add-crud lst)
Construye lista de configuraciones de funciones CRUD de una tabla en especifico. Retorna lista de listas.

crud
modf-crud-aux :: string string (listof list?) → void?
(modf-crud-aux schema ntable lstcrud)
Modifica lista de configuraciones de funciones CRUD, elimina y agrega nuevas configuraciones de una tabla en especifico. Retorna void?.

crud
modf-crud :: string string (listof list?) → void?
(modf-crud schema ntable lstcrud)
Modifica lista de configuraciones de funciones CRUD, si estructura vcrudg se encuentra vacía, se agregan las configuraciones, de lo contrario se modifican. Retorna void?.

crud
add-crud-per :: (listof list?) → (listof list?)
(add-crud-per lst)
Construye lista de configuraciones de funciones CRUD personalizadas de una tabla en especifico. Retorna lista de listas.

crud

modf-crud-per-aux :: string string (listof list?) → void?
(modf-crud-per-aux schema ntable lstcrud)
Modifica lista de configuración de funciones CRUD personalizadas. Retorna void?.

crud
modf-crud-per :: string string (listof list?) → void?
(modf-crud-per schema ntable lstcrud)
Modifica lista de configuraciones de funciones CRUD personalizadas, si lista vcrudg-per se encuentra vacía, se agregan las configuraciones, de lo contrario se modifican. Retorna void?.

crud
filter-crud :: string (listof list?) → (listof list?)
(filter-crud ntable lst)
Obtiene lista de configuraciones de funciones CRUD de una tabla en especifico. Retorna lista de listas.

crud
render :: (listof list?) → list?
(render lst)
Construye lista que representa el cuerpo de tabla dinámica de funciones CRUD para una tabla en especifico. Retorna lista.

crud
render-crud-table :: string → list?
(render-crud-table ntable)
Construye cuerpo de tabla dinámica de funciones CRUD para una tabla en especifico. Retorna lista.

crud
crud-per-text-aux :: string (listof list?) → string
(crud-per-text-aux str lst)
Convierte lista de configuraciones de funciones CRUD personalizadas a cadena de texto separadas por delimitador %. Retorna cadena de texto.

crud

crud-per-text :: string → string
(crud-per-text)
Filtra lista de configuraciones de funciones CRUD personalizadas por una tabla en especifico, y convierte dicha lista en cadena de texto. Retorna cadena de texto.

crud
format-crud-data :: nothing → void?
(format-crud-data)
Construye formato adecuado para cada función CRUD de la totalidad de las tablas. Retorna lista de listas.

crud
format-crud-per-data :: nothing → (listof list?)
(format-crud-per-data)
Construye lista de listas con formato adecuado para cada función CRUD personalizada. Retorna lista de listas.

crud
crud-exist :: nothing → boolean
(crud-exist)
Verifica si existen configuraciones de funciones CRUD. Retorna #t si existen, #f en caso contrario.

crud
crud-per-exist :: nothing → boolean
(crud-per-exist)
Verifica si existen configuraciones de funciones CRUD personalizadas. Retorna #t si existen, #f en caso contrario.

crud
clean-crud-data :: nothing → void?
(clean-crud-data)
Elimina datos almacenados en estructura vcrudg. Retorna void?.

crud
remove-only-table :: string → void?

(remove-only-table str)
Elimina configuraciones de funciones CRUD asociadas a una tabla en especifico. Retorna void?.

crud
remove-only-table-per :: string → void?
(remove-only-table-per str)
Elimina configuraciones de funciones CRUD personalizadas asociadas a una tabla en especifico. Retorna void?.

3.8. Funciones en módulo generador

generador
make-page-view :: string string string → expr?
(make-page-view id sch ntb)
Despliega visualmente configuración de campos de una tabla del esquema. Retorna expresión.

generador
make-page :: syntax → syntax
(make-page sch npg ntb)
Macro que se extiende en su definición para generar función encargada de especificar y crear página web de una tabla en especifico. Usa en el proceso la configuración hecha a los campos y funciones CRUD. Retorna sintaxis.

generador
render-page :: string string string → expr?
(render-page sch npg ntb)
Construye página web de una tabla en especifico. Retorna expresión.

generador
menu-lst-aux :: string → expr?
(menu-lst-aux elemt)
Crea elemento HTML li para cada elemento de la lista. Retorna expresión.

generador
menu-lst :: list? → expr?
(menu-lst lst)
Crea elemento HTML ul con lista de elementos li. Retorna expresión.

generador
response-msg :: nothing → expr?
(response-msg)
Construye expresión para la muestra de mensajes generados por la aplicación prototipo. Retorna expresión.

generador
form-render-real :: string → expr?
(form-render-real ntable)
Construye formulario web para tabla especificada. Su construcción se basa en la configuración hecha en sus campos. Retorna expresión.

generador
get-checkbox :: string list? → list?
(get-checkbox id opcs)
Construye lista con elementos HTML checkbox, que fueron especificados en la tabla de configuración de campos. Retorna lista.

generador
get-radio :: string list? → list?
(get-radio id opcs)
Construye lista con elementos HTML radio, que fueron especificados en la tabla de configuración de campos. Retorna lista.

generador
get-textarea :: string number/string → expr?
(get-textarea id sizef)
Construye elemento HTML textarea. Retorna expresión.

generador
get-select :: list? → list?
(get-select opcs)
Construye lista con elementos option para elemento HTML select que fueron especificados en la tabla de configuración de campos. Retorna lista.

generador
get-select-date :: string → expr?
(get-select-date id)
Construye elementos HTML select para campo tipo select_date, especificado en tabla de configuración de campos. Retorna expresión.

generador

get-option-day :: nothing → list?
(get-option-day)
Construye lista de elementos HTML option para dias. Retorna lista.

generador
get-option-month :: nothing → list?
(get-option-month)
Construye lista de elementos HTML option para meses. Retorna lista.

generador
get-option-year :: nothing → list?
(get-option-year)
Construye lista de elementos HTML option para años. Retorna lista.

generador
get-list-num :: number number → list?
(get-list-num a b)
Construye lista de números en el rango especificado. Retorna lista.

generador
get-month :: number → string
(get-month i)
Consulta mes ubicado en posición especifica. Retorna string.

generador
get-month-aux :: number → string
(get-month-aux lst i idx)
Busca mes ubicado en posición especifica. Retorna string.

generador
ns :: number → string
(ns i)
Convierte entero en cadena de texto. Retorna string.

generador

get-elemts-table :: string (listof list?) → (listof list?)
(get-elemts-table ntable lst)
Consulta configuración de campos de tabla especificada. Retorna lista de listas.

generador
rc :: string → string
(rc str)
Remueve comillas dobles de cadena de texto especificada. Retorna string.

generador
to-string-func :: expr? → string
(to-string-func f)
Convierte expresión en cadena de texto. Retorna string.

generador
create-app-descfile :: string → output-port?
(create-app-descfile file)
Crea descriptor de archivo para escribir el código generado. El nombre del archivo que se relaciona con el descriptor es schema-app.rkt. Retorna descriptor.

generador
write-cv-req :: string → string
(write-cv-req schema)
Especifica los requerimientos para manipulación de funciones CRUD y de vista. Retorna string.

generador
write-reqs :: output-port? string → void?
(write-reqs df schema)
Escribe los requerimientos de la aplicación prototipo en archivo schema-app.rkt. Retorna void?.

generador
defv :: string list? → string
(defv schema lst)
Especifica reglas de despacho de aplicación prototipo. Retorna string.

generador
deff :: string list? → string
(deff schema lst)
Especifica funciones de despacho de aplicación prototipo. Retorna string.

generador
write-rules :: output-port? string string list? → void?
(write-rules df schema lst)
Escribe reglas y funciones de despacho en archivo schema-app.rkt. Retorna void?.

generador
write-pages :: output-port? string (listof list?) → void?
(write-pages df schema lst)
Escribe las páginas que representan a cada una de las tablas del esquema de base de datos que fueron activadas, considerando la configuración de sus campos y funciones CRUD asociadas, en archivo schema-app.rkt. Usa macro make-page para extender las configuraciones de los campos y funciones CRUD. Retorna void?.

generador
write-server :: output-port? string list? → void?
(write-server df schema lst)
Escribe configuración de servidor Web en archivo schema-app.rkt. Que permite despliegue y ejecución de aplicación prototipo. Retorna void?.

generador
create-dirs-aux :: string string → boolean
(create-dirs-aux dir dirh)
Crea directorios de aplicación prototipo. Directorio raiz y de archivos estáticos htdocs. Retorna #t si fueron creados, #f en caso contrario.

generador
create-app-file :: schema list? → void?
(create-app-file schema lst)
Inicia creación de archivo de código fuente de aplicación prototipo en base a la configuración de los campos y funciones CRUD de la totalidad de las tablas. (Páginas y formularios). Retorna void?.

generador
create-crud-file :: string → void?
(create-crud-file schema)
Inicia creación de archivo de código fuente de aplicación prototipo en base a la configuración de las funciones CRUD. (Funciones de persistencia). Retorna void?. En caso de no existir funciones CRUD, el archivo es eliminado.

generador
create-view-file :: string → void?
(create-view-file schema)
Inicia creación de archivo de código fuente de aplicación prototipo con funciones de vista. Retorna void?. En caso de no existir funciones CRUD, el archivo es eliminado.

3.9. Funciones en módulo tablas

tablas
inic-vtablas :: string → void?
(inic-vtablas schema)
Construye lista de tablas pertenecientes a un esquema. El nombre del esquema lo adiciona como primer elemento. Retorna lista.

tablas
get-vtablas :: string → list?
(get-vtablas schema)
Consulta tablas pertenecientes al esquema especificado y las asigna a estructura vtablasg. Retorna lista.

tablas
check-ext-table :: string → boolean
(check-ext-table ntable)
Verifica si tabla especificada se encuentra listada en estructura vtablas. Retorna #t si es así, #f en caso contrario.

tablas
check-table :: list? string → boolean
(check-table lst ntable)
Busca si tabla especificada se encuentra listada en estructura vtablas. Retorna #t si es así, #f en caso contrario.

tablas
active-table :: string → string
(active-table ntable)
Verifica si tabla especificada se encuentra activa, es decir se encuentra en estructura vtablas. Retorna string.

tablas
add-table :: string → list?
(add-table ntable)
Agrega tabla especificada en estructura vtablas. Retorna lista.

tablas
remove-table :: list? string → list?
(remove-table lst ntable)
Elimina tabla especificada de estructura vtablas. Retorna lista.

tablas
modf-tables :: string → void?
(modf-tables tablevalue)
Verifica si usuario activó alguna tabla del esquema (opción on), si es así, almacena su nombre en estructura vtablas, de lo contrario la elimina. Retorna void?

tablas
get-tablas-alm :: nothing → list?
(get-tablas-alm)
Retorna lista de tablas.

tablas
get-tables :: nothing → list?
(get-tables)
Retorna lista de tablas activas.

tablas
clean-tablas-data :: nothing → void?
(clean-tablas-data)
Elimina datos almacenados en estructuras vtablasg y vtablas. Retorna void?.

3.10. Funciones en módulo view-funcs

view-funcs
save-view-funcs-file :: string → void?
(save-view-funcs-file schema)
Crea archivo de código fuente con funciones de vista. El nombre para el archivo es schema-view.rkt. Retorna void?.

view-funcs
remove-view-funcs-file :: string → void?
(remove-view-funcs-file schema)
Elimina archivo schema-view.rkt de código fuente de funciones de vista. Retorna void?.

view-funcs
get-render-table :: nothing → expr?
(get-render-table)
Crea expresión encargada de crear tabla en HTML. Retorna expresión.

view-funcs
get-render-table-aux :: nothing → expr?
(get-render-table-aux)
Función auxiliar que crea expresión para construir cuerpo de tabla HTML. Retorna expresión.

view-funcs
get-sql-date-str :: nothing → expr?
(get-sql-date-str)
Crea expresión encargada de convertir estructura de fecha (date) sql-date? o valor numérico en cadena de texto. Retorna expresión.

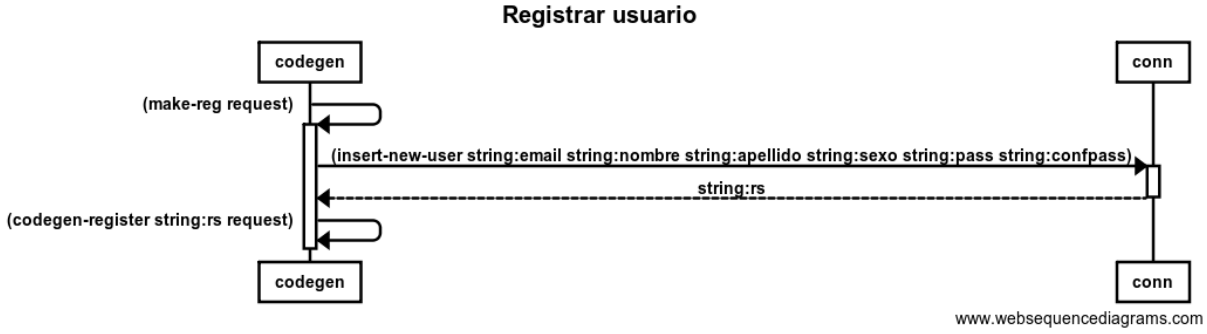
view-funcs
write-head-view :: open-output-file → void?
(write-head-view df)
Escribe en archivo schema-view.rkt funciones de vista. Retona void?.

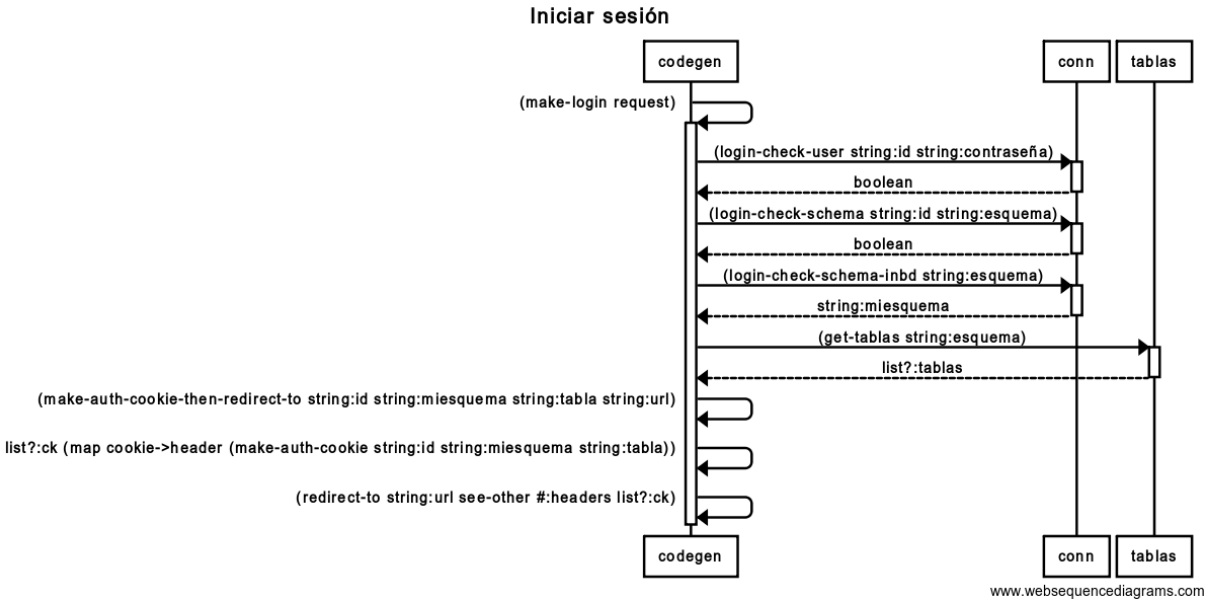
view-funcs
write-end-view :: open-output-file → void?
(write-end-view df)
Escribe en archivo schema-view.rkt función de provisión de funciones. Retorna void?.

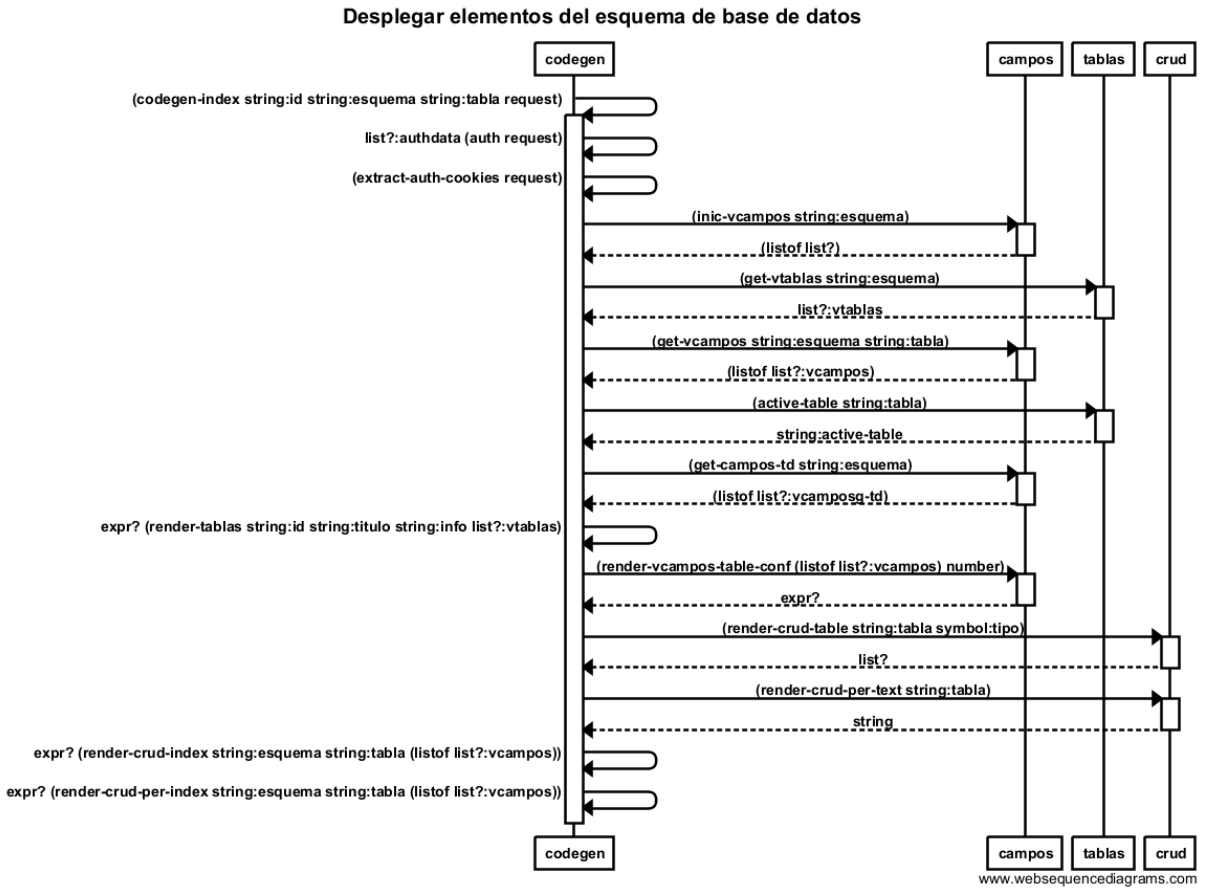
3.11. Diagramas de secuencia

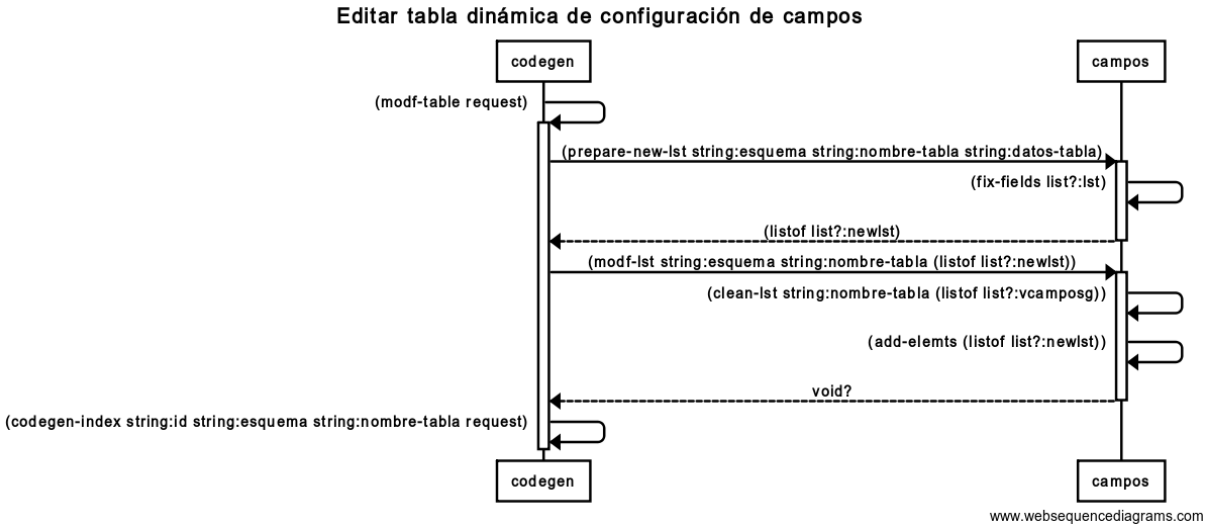
Se presenta a continuación los diagramas de secuencia para los siguientes requerimientos:

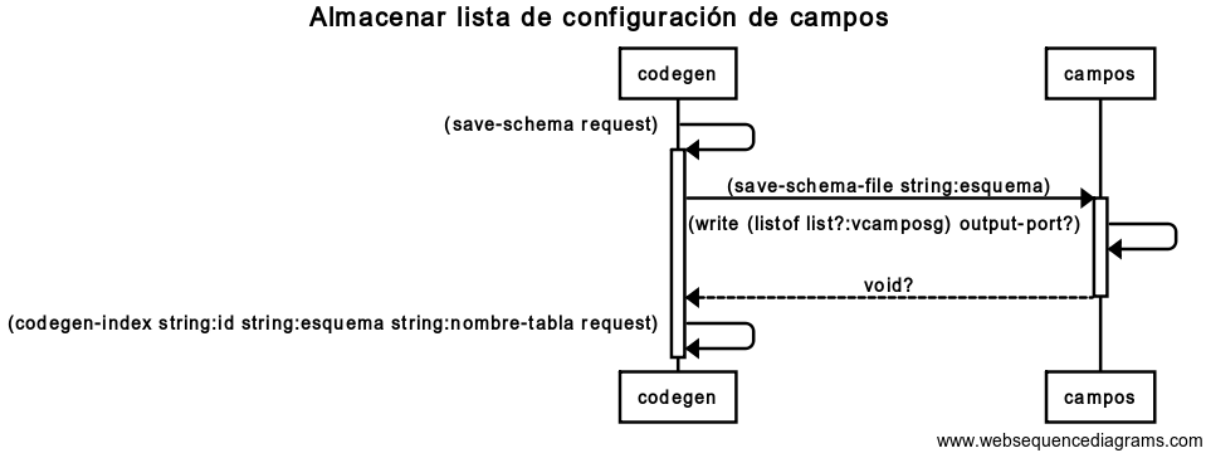
- Registrar usuario
- Iniciar sesión
- Desplegar elementos del esquema de base de datos
- Editar tabla dinámica de configuración de campos
- Almacenar lista de configuración de campos
- Almacenar lista de configuración de funciones CRUD
- Almacenar lista de configuración de funciones CRUD personalizadas
- Visualizar gráficamente configuración de campos
- Exportar aplicación

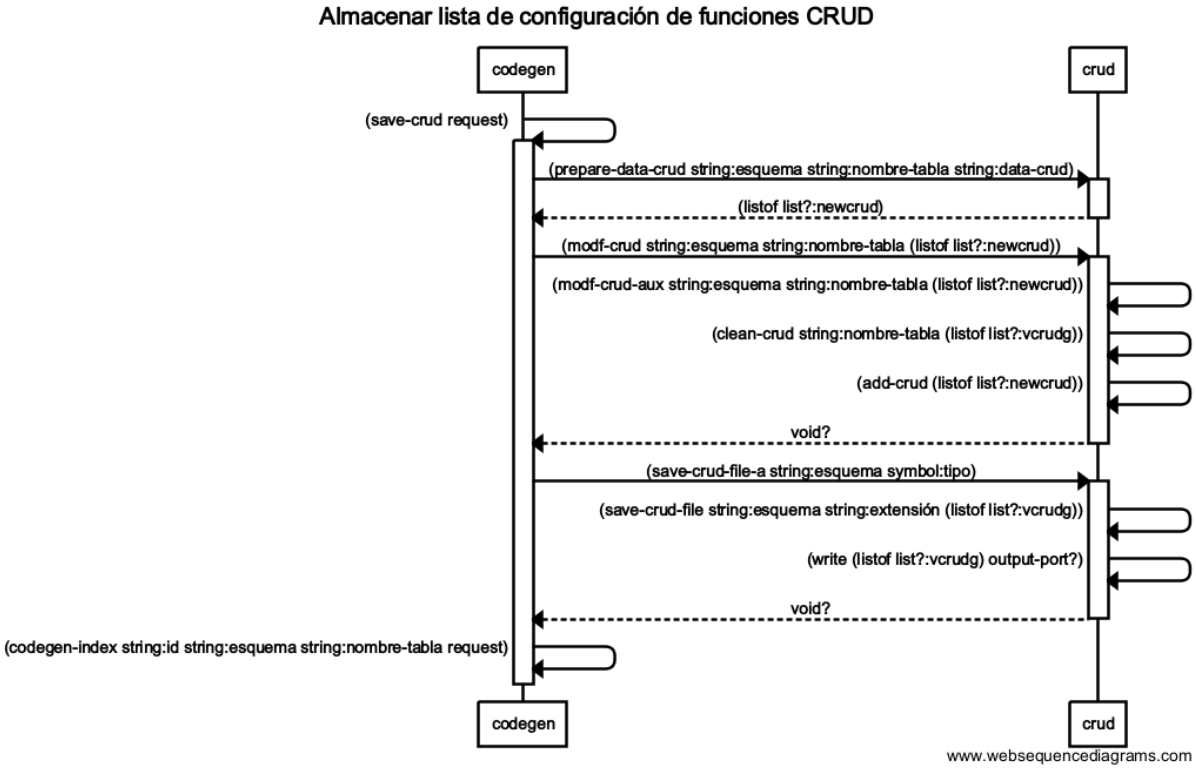


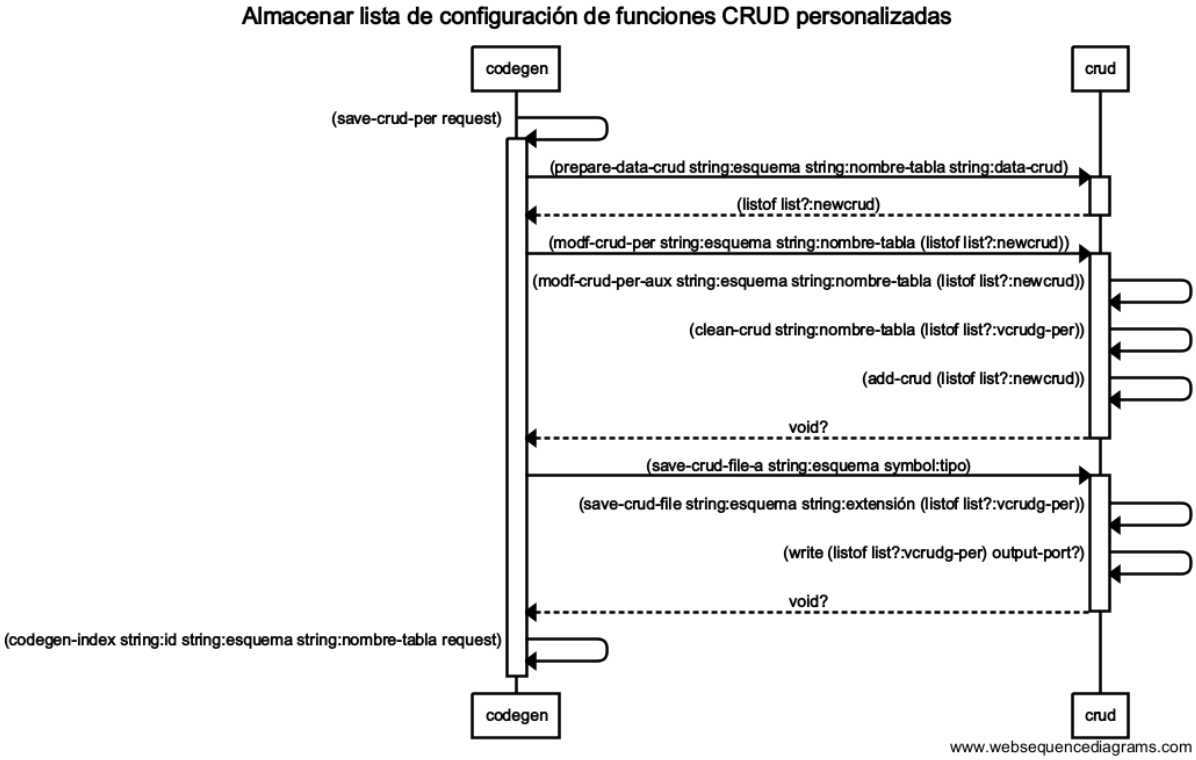

Iniciar sesión
codegen
conn
tablas
(make-login request)
(login-check-user string:id string:contraseña)
boolean
(login-check-schema string:id string:esquema)
boolean
(login-check-schema-inbd string:esquema)
string:miesquema
(get-tablas string:esquema)
list?:tablas
(make-auth-cookie-then-redirect-to string:id string:miesquema string:tabla string:url)
list?:ck (map cookie->header (make-auth-cookie string:id string:miesquema string:tabla))
(redirect-to string:url see-other #:headers list?:ck)
codegen
conn
tablas
www.websequencediagrams.com

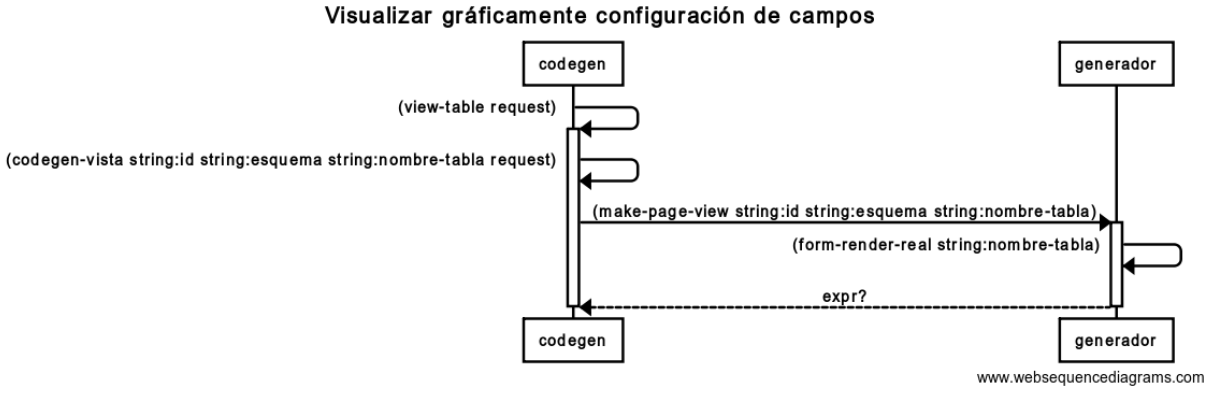

Desplegar elementos del esquema de base de datos
codegen
campos
tablas
crud
(codegen-index string:id string:esquema string:tabla request)
list?:authdata (auth request)
(extract-auth-cookies request)
(inic-vcampos string:esquema)
(listof list?)
(get-vtablas string:esquema)
list?:vtablas
(get-vcampos string:esquema string:tabla)
(listof list?:vcampos)
(active-table string:tabla)
string:active-table
(get-campos-td string:esquema)
(listof list?:vcamposq-td)
expr? (render-tablas string:id string:titulo string:info list?:vtablas)
(render-vcampos-table-conf (listof list?:vcampos) number)
expr?
(render-crud-table string:tabla symbol:tipo)
list?
(render-crud-per-text string:tabla)
string
expr? (render-crud-index string:esquema string:tabla (listof list?:vcampos))
expr? (render-crud-per-index string:esquema string:tabla (listof list?:vcampos))
codegen
campos
tablas
crud
www.websequencediagrams.com

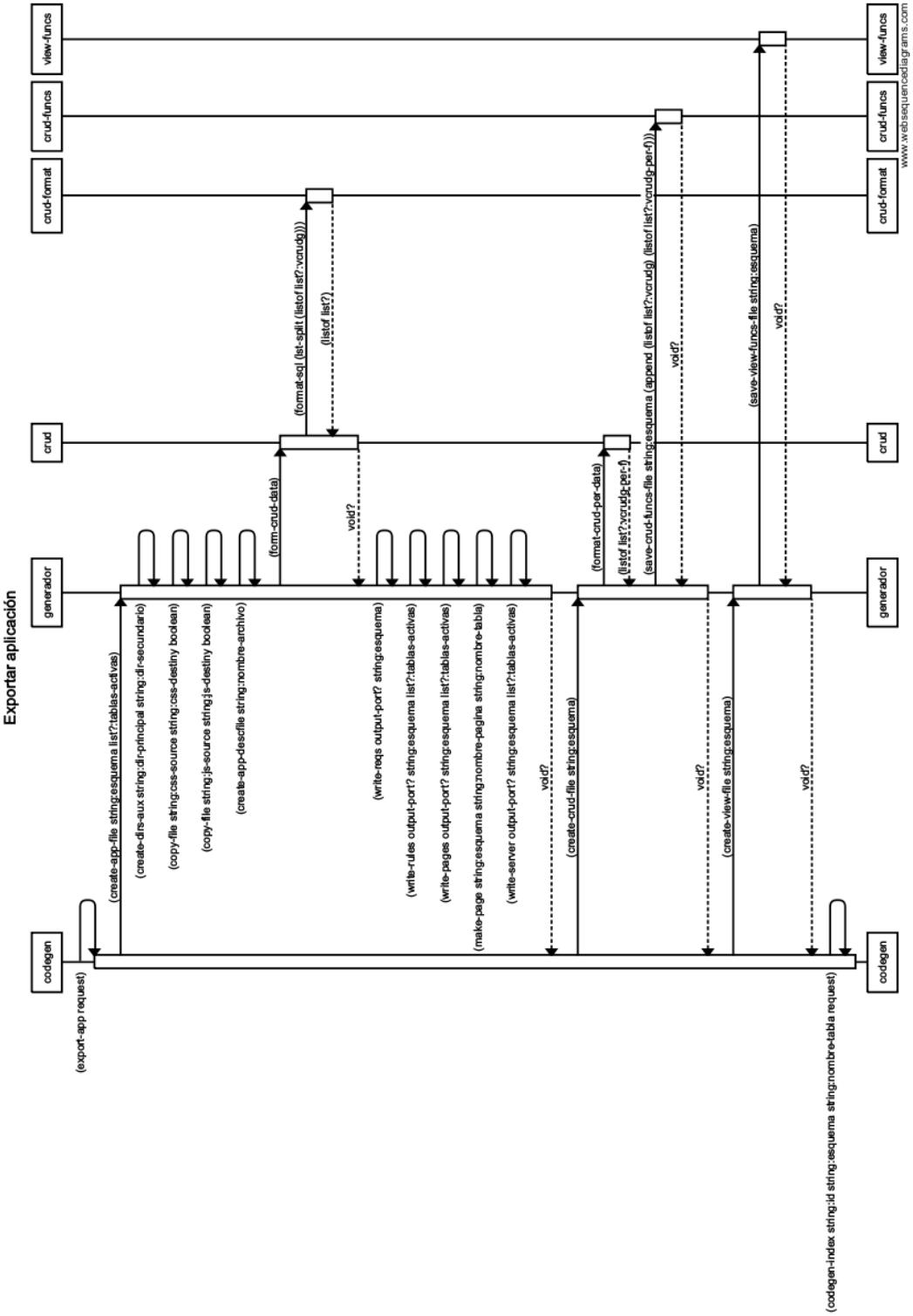

Editar tabla dinámica de configuración de campos
codegen
campos
(modf-table request)
(prepare-new-lst string:esquema string:nombre-tabla string:datos-tabla)
(fix-fields list?:lst)
(listof list?:newlst)
(modf-lst string:esquema string:nombre-tabla (listof list?:newlst))
(clean-lst string:nombre-tabla (listof list?:vcamposg))
(add-elemts (listof list?:newlst))
void?
(codegen-index string:id string:esquema string:nombre-tabla request)
codegen
campos
www.websequencediagrams.com


Almacenar lista de configuración de campos
codegen
campos
(save-schema request)
(save-schema-file string:esquema)
(write (listof list?:vcamposg) output-port?)
void?
(codegen-index string:id string:esquema string:nombre-tabla request)
codegen
campos
www.websequencediagrams.com


Almacenar lista de configuración de funciones CRUD
codegen
crud
(save-crud request)
(prepare-data-crud string:esquema string:nombre-tabla string:data-crud)
(listof list?:newcrud)
(modf-crud string:esquema string:nombre-tabla (listof list?:newcrud))
(modf-crud-aux string:esquema string:nombre-tabla (listof list?:newcrud))
(clean-crud string:nombre-tabla (listof list?:vcrudg))
(add-crud (listof list?:newcrud))
void?
(save-crud-file-a string:esquema symbol:tipo)
(save-crud-file string:esquema string:extensión (listof list?:vcrudg))
(write (listof list?:vcrudg) output-port?)
void?
(codegen-index string:id string:esquema string:nombre-tabla request)
codegen
crud
www.websequencediagrams.com


Almacenar lista de configuración de funciones CRUD personalizadas
codegen
crud
(save-crud-per request)
(prepare-data-crud string:esquema string:nombre-tabla string:data-crud)
(listof list?:newcrud)
(modf-crud-per string:esquema string:nombre-tabla (listof list?:newcrud))
(modf-crud-per-aux string:esquema string:nombre-tabla (listof list?:newcrud))
(clean-crud string:nombre-tabla (listof list?:vcrudg-per))
(add-crud (listof list?:newcrud))
void?
(save-crud-file-a string:esquema symbol:tipo)
(save-crud-file string:esquema string:extensión (listof list?:vcrudg-per))
(write (listof list?:vcrudg-per) output-port?)
void?
(codegen-index string:id string:esquema string:nombre-tabla request)
codegen
crud
www.websequencediagrams.com


Visualizar gráficamente configuración de campos
codegen
generador
(view-table request)
(codegen-vista string:id string:esquema string:nombre-tabla request)
(make-page-view string:id string:esquema string:nombre-tabla)
(form-render-real string:nombre-tabla)
expr?
codegen
generador
www.websequencediagrams.com


Exportar aplicación
codegen
generador
crud
crud-format
crud-funcs
view-funcs
(export-app request)
(create-app-file string:esquema list?:tablas-activas)
(create-dirs-aux string:dir-principal string:dir-secundario)
(copy-file string:css-source string:css-destiny boolean)
(copy-file string:js-source string:js-destiny boolean)
(create-app-descfile string:nombre-archivo)
(form-crud-data)
(format-sql (lst-split (listof list?:vcrudg)))
(listof list?)
void?
(write-reqs output-port? string:esquema)
(write-rules output-port? string:esquema list?:tablas-activas)
(write-pages output-port? string:esquema list?:tablas-activas)
(make-page string:esquema string:nombre-pagina string:nombre-tabla)
(write-server output-port? string:esquema list?:tablas-activas)
void?
(create-crud-file string:esquema)
(format-crud-per-data)
(listof list?:vcrudg-per-f)
(save-crud-funcs-file string:esquema (append (listof list?:vcrudg) (listof list?:vcrudg-per-f)))
void?
void?
(create-view-file string:esquema)
(save-view-funcs-file string:esquema)
void?
void?
(codegen-index string:id string:esquema string:nombre-tabla request)
www.websequencediagrams.com