Tabla de contenido

INTRODUCCIÓN

El presente documento detalla las experiencias, objetivos, resultados y demás documentación del proyecto denominado G*eneración de operaciones de acceso de datos CRUD en función del modelo para el Framework de desarrollo web Yesod* (Yesod-CRUD), desarrollado como tesis de grado en la Escuela Colombiana de Ingeniería Julio Garavito ,con el objetivo de facilitar el desarrollo de aplicaciones web en el Framework Yesod, mediante la generación automática de la lógica e interfaz de usuario requerida para, crear, modificar, eliminar y listar; e incentivar de este modo el uso de este Framework exponiendo los beneficios que ofrece.

El proyecto está dirigido a todas las personas que actualmente utilizan Yesod de manera comercial o académica, con el fin de ofrecerles una nueva ventaja a la hora de desarrollar sus aplicaciones. Dicha comunidad que puede ser pequeña numéricamente, pero cuenta con una alta productividad y crecimiento gracias a la política casi absoluta de software libre. Es por ello que este proyecto se deja disponible en un repositorio público GitHub con licencia MIT para que cualquier persona pueda aportar a su juicio. Aun así, se propone un plan de continuidad basado en las experiencias obtenidas desarrollando el paquete y como usuario de Yesod.

Dentro de la Escuela Colombiana de Ingeniería Julio Garavito, el proyecto busca ser una referencia para futuros proyectos (continuación o no de este) que se enfoquen en la programación funcional, generación de código, verificación de código, DSL´s, o cualquier otro tema directa o indirectamente planteado en este proyecto.

Este documento ofrece además una introducción teórica a Yesod definiendo sus características, basado en la documentación existente y expresada de forma muy concisa. Sin embargo, este documento no es una guía o tutorial acerca de cómo programar en Yesod o cómo usar las librerías de generación de código, ya que se limita a definir los conceptos técnicos usados, dejando a interés del lector una investigación más profunda en estos temas con ayuda de la bibliografía incluida en el documento. No obstante si se incluye una guía y un ejemplo de cómo utilizar el paquete desarrollado en el proyecto, explicando las formas de uso, requerimientos de la aplicación, limitaciones del paquete y errores comunes de su uso.

Generación de operaciones de acceso de datos CRUD en función del modelo para el Framework de desarrollo web Yesod (Yesod-CRUD)

PARTE I

1. Resumen ejecutivo

Desarrollar un paquete para el Framework de desarrollo web Yesod de tal forma que sea posible generar las operaciones de acceso de datos (Crear, Modificar, Eliminar y Consultar) junto a sus respectivas vistas, a partir del modelo de datos y persistencia presente en las aplicaciones desarrolladas en este Framework.

2. Planteamiento del problema

A partir del modelo de datos se puede automatizar algunas tareas que anteriormente eran desarrolladas de forma manual. Por ejemplo, en el Framework Yesod algunas de estas tareas han sido automatizadas como el manejo de la persistencia y la creación de los tipos de datos utilizados dentro de una aplicación.

Es por esta razón que se plantea automatizar el manejo de los datos de una aplicación, conocido como CRUD, a partir del modelo de datos. Parte del estudio inicial consiste en la revisión de las buenas prácticas arquitecturales y de ingeniería de software para lograr un paquete coherente garantizando el desacoplamiento de las tres capas del modelo MVC según el estándar de Yesod

3. Justificación

Los tiempos de desarrollo son en la actualidad un factor clave a la hora de embarcarse en nuevos desarrollos de aplicaciones web, sin embargo en muchos lenguajes y frameworks se utiliza gran parte del tiempo en la construcción de las funciones de CRUD, lo que hace que se invierta tiempo y recurso humano en partes del desarrollo que puede ser generado de manera automática o semiautomática. Se debe tener en cuenta la versatilidad y usabilidad a la hora de automatizar los CRUD de manera que permita ser utilizado en la mayoría de aplicaciones sin comprometer los diseños y funcionalidades particulares. Una de las ventajas que ofrece el framework Yesod tanto el lenguaje de base Haskell es llevar la mayor cantidad de errores a tiempo de compilación. Se pretende incorporar esta filosofía también al paquete propuesto.

4. Objetivo general

Desarrollar un paquete sobre el framework de desarrollo web Yesod de tal forma que sea posible la generación de operaciones de acceso de datos CRUD a partir del modelo de datos de una aplicación.

5. Objetivos específicos

- Desarrollo de un paquete para el framework Yesod a fin de que sea posible generar las operaciones de CRUD.
- Ofrecer una generación versátil, es decir que el desarrollador pueda seleccionar cuáles operaciones desea para cada entidad definida.
- Generar de manera automática las vistas necesarias para cada una de las operaciones (Formularios para la creación y modificación, listas para consultar y eliminar).
- Configurar una máquina virtual con todos los recursos necesarios para el uso del paquete que sirva además como ambiente de desarrollo para extensiones futuras.
- Generar una documentación para permitir que dicha extensionalidad pueda ser usada en desarrollos futuros

PARTE II: Estado del arte

1. Precedentes en otros Frameworks

- Synfony: es un Framework web de código abierto para PHP (Requiere PHP 5.3) basado en el modelo MVC, encaminado a reducir el tiempo de desarrollo de una aplicación web compatible con la mayoría de gestores de base de datos, cuenta con un sistema para la generación automática de código CRUD en función de su modelo de datos.
- Yii: Framework web orientado a objetos de código abierto para PHP basado en el modelo MVC, utiliza el patrón DAO para la migración de la base de datos, incluye un asistente para la generación automática de código llamada Gii, que permite parametrizar cosas tales como las validaciones para los formularios dentro de la generación de código.
- Play es un Framework web de código abierto para Scala y JAVA basado en el modelo MVC, al igual que Synfony apunta a la productividad del desarrollador y busca reducir tiempos, Incluye un módulo para la generación del CRUD junto a sus interfaces, generando vista del formulario y una vista para listar, permite además configurar o extender el código generado, añadiendo nuevas validaciones, personalizando la interfaz entre otros
- Yesod Helpers Crud: Librería desarrollada para la generación de CRUD para el Framework web de Yesod hasta su versión 0.4.1, en su documentación se expone funcionalidad para 4 servicios, listar, crear,

modificar y eliminar, para ello utiliza métodos GET para listar y métodos GET y POST para todos los demás. Este fue un primer intento para la generación de CRUD, desarrollado para la versión 0.4.0.3 y dejado de lado dados los cambios y evolución que tuvo el Framework, el cual en este momento se encuentra en la versión 1.4. Esa versión de CRUD no ofrece la internacionalización ni esquemas de autorización y autenticación, adicionales a los diferentes cambios introducidos en la creación de subsistíos, formas y en la capa de persistencia en su versión 2. Esta librería es obsoleta aunque sirve como base o referencia para el desarrollo del proyecto.

2. ¿Qué es Yesod?

Yesod es un Framework de desarrollo web para el lenguaje Haskell que ofrece grandes ventajas a la hora de realizar desarrollos web, entre las más destacadas están:

- **"Migración de errores de tiempo de ejecución a tiempo de compilación":** Gracias a la verificación de tipos Yesod buscar aplicar una filosofía a sus desarrollos, "en general si el código compila funciona", de esta manera se logra disminuir significativamente el tiempo de desarrollo y aumenta la calidad de las aplicaciones.

- "**Escalable y eficiente":** Gracias a Yesod podemos escribir código simple pero eficiente y con gran rendimiento, sus bibliotecas proporcionan una interfaz agradable y segura mientras consiguen un rendimiento similar al de lenguajes como C con acceso directo a la memoria.

- **"Código altamente funcional":** El desarrollo web puede ser altamente repetitivo en términos de su creación y con una gran cantidad de líneas de código, Yesod es diferente ya que permite obtener funcionalidades completas en pocas líneas de código y con su verificación en tiempo de compilación de que el código es correcto.[6]

3. Primeros pasos en Yesod

En la guía "Primeros pasos en Yesod" [7] podrá encontrar el paso a paso para crear un proyecto en Yesod

4. Elementos de un proyecto web en Yesod

4.1 Persistencia

Un elemento clave a analizar al momento de realizar una aplicación web es la capa de persistencia y el mecanismo para realizar la comunicación entre ella y la

capa de aplicación. Se tiene por ejemplo en los lenguajes orientados a objetos los ORM (object relational mapping) [8] como Hibernate para JAVA; para el caso de Yesod la solución se llama ***Persistent.*** El cual permite sincronizarse con diferentes medios de almacenamiento utilizando una interfaz de consulta segura, concisa, declarativa y manteniendo la política de tipos de datos seguros de Haskell. Algunas de sus principales características son:

- Conexiones para PostgreSQL, SQLite, MySQL y MongoDB, con soporte experimental para Redis.
- Modelado de datos sencillo que permite definir relaciones manteniendo tipos de datos seguros.
- Migración automática a la base de datos [4].

4.2 Vista

Un segundo elemento es la interfaz de la aplicación, tradicionalmente en la gran mayoría de Frameworks utilizan HTML, CSS y JavaScript, para el caso de Yesod la respuesta es ***Shakespearean Templates,*** una familia de "template languages" para la creación de los HTML, CSS y JavaScript. Algunos principios de Shakespearean son:

- Garantiza un código bien formado en tiempo de compilación.
- Tipos estáticos seguros.
- Validación automática de interpolaciones, gracias a las URL's de tipo seguro.

Está compuesto por cuatro "template languages", estos son:

- Hamlet para HTML: HTML pero sin tags de cierre en su lugar usa la indentación para verificar el código, con interpolación
- Lucius para CSS: la sintaxis de CSS pero soporta interpolación
- Cassius para CSS: similar a Lucius pero sin punto y comas (;) o corchetes ({,}), utiliza la indentación para verificar el código
- Julius para JavaScript]: JavaScript pero con interpolación [10].

4.3 Widgets

Al igual que ocurre con la capa de persistencia que requiere definir un medio de comunicación entre la persistencia y la aplicación, es necesario definir a su vez un medio de interacción entre la aplicación y la vista, Shakespearean Templates resuelve en gran parte este problema pero no lo hace solo, se apoya en un segundo elemento muy importante llamado ***Widgets,*** estos sin embargo, no solo permiten dicha comunicación sino que ofrecen una estructura que permite integrar los cuatro elementos que componen Shakespearean[11].

4.4 Forms

Uno de los problemas más frecuentes en las aplicaciones web es la captura y validación de los datos ingresados por los usuarios, Yesod resuelve este y muchos problemas más gracias al uso de las denominadas ***Forms,*** algunos de sus beneficios son:

- Validación de datos.
- Conversión de las entradas de texto en tipos de datos Haskell al momento de enviar el formulario.
- Generación de código HTML para la forma.
- Asignación de nombres automática.
- Conexión con formas previamente creadas.
- Verificación de tipos de la forma en tiempo de compilación.

Para lograr todo esto Yesod posee 3 tipos de Forms:

- Input Form: Reciben solamente la entrada sin generar código HTML, ideal para formularios que ya están creados.
- Applicative Form: Es la forma más usada debido a su facilidad, ya que tiene una forma declarativa.
- Monadic Form: Es un estilo de forma mucho más flexible y poderosa que la aplicativa, aunque es más verbosa, es decir requiere una mayor cantidad de código. [12]

4.5 Routing

En Yesod las rutas se encuentran centralizadas en un único archivo similar a DJango o GAE pero a diferencia de estos, Yesod define un tipo de dato intermedio creando funciones de conversión en las dos vías. Para ello Yesod define un DSL (Domain specific language) y mediante Haskell Template convierte estas funciones en código Haskell. [13]

4.6 Handlers

Finalmente encontramos lo que junto a las rutas compondría el controlador en un modelo MVC (Modelo Vista controlador), estos son los ***Handlers***, en ellos encontramos los métodos que resuelven las peticiones generadas a partir de una ruta; Estas funciones reciben como parámetro aquellos definidos por la ruta definida, esto ofrece una validación de tipos y permite mover gran cantidad de errores de tiempo de ejecución a tiempo de ejecución y ofrecer rutas de tipo seguro.

4.7 SubSites, Authentication y otras características de Yesod:

Los elementos anteriormente mencionados describen los componentes básico presentes en prácticamente todas las aplicaciones desarrolladas en Yesod, sin embargo existen componentes adicionales tales como la Autenticación de usuarios o los SubSitios, que no se mencionan en detalle pero que ayudan a potencializar las aplicaciones, facilitan la estandarización de código entre otras.

5. Despliegue

Todas las aplicaciones web deben tener un servidor de despliegue, y en el caso de Yesod este servidor se llama ***Keter;*** éste ofrece las siguientes acciones:

- Se conecta al puerto principal(generalmente el puerto 80)
- Lanza automáticamente las aplicaciones y realiza un monitoreo de procesos, de modo tal que pueda relanzar automáticamente en caso de que alguna caiga
- Gestión de archivos[10]

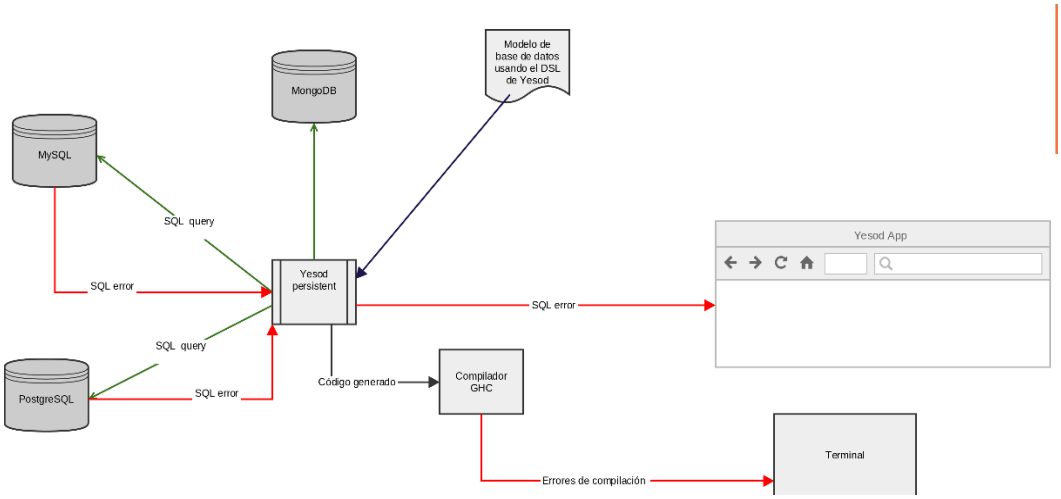
Para ello todos los proyectos en Yesod incluyen un archivo keter.yml el cual contiene toda la información requerida para el despliegue, es posible además definir dentro de la configuración el servidor de despliegue de forma que con el comando *Yesod Keter* se conecte automáticamente al servidor y realice el despliegue.

Existe sin embargo una restricciones al momento de hacer despliegues en Keter, ya que este no soporta despliegue sobre una dirección IP, dicho de otra forma la aplicación debe tener un dominio DNS para que pueda ser desplegado.

6. Diagramas

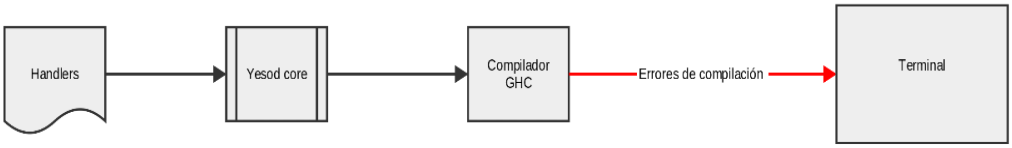
6.1 Persistencia:

El siguiente gráfico es un diagrama ilustra la capa de persistencia en el Framework Yesod.



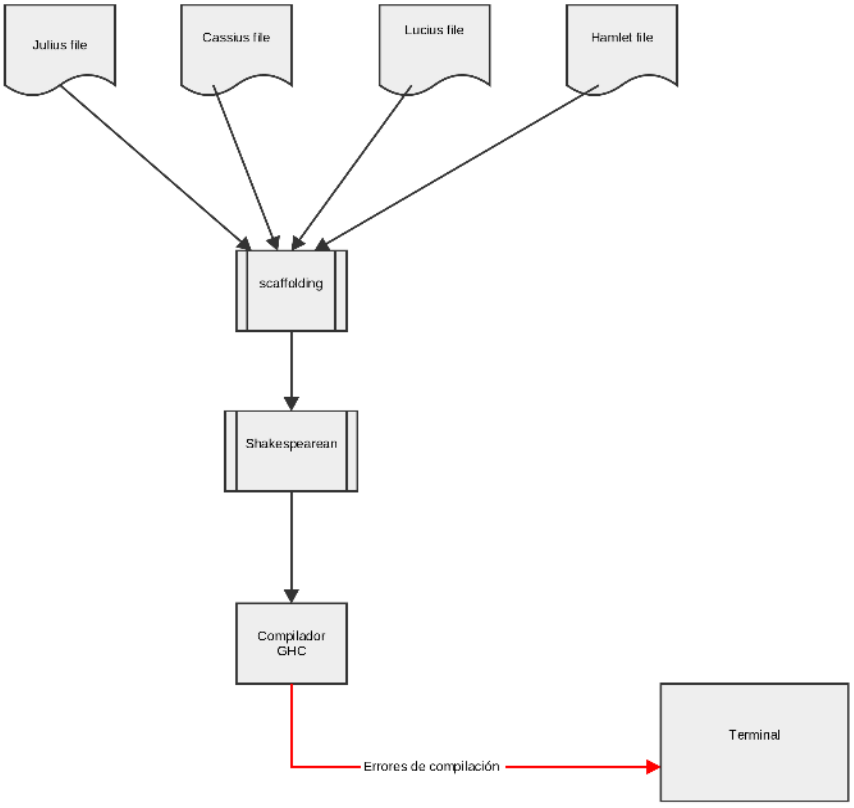
6.2 Lógica:

El siguiente gráfico es un diagrama ilustra la capa lógica en el Framework Yesod.



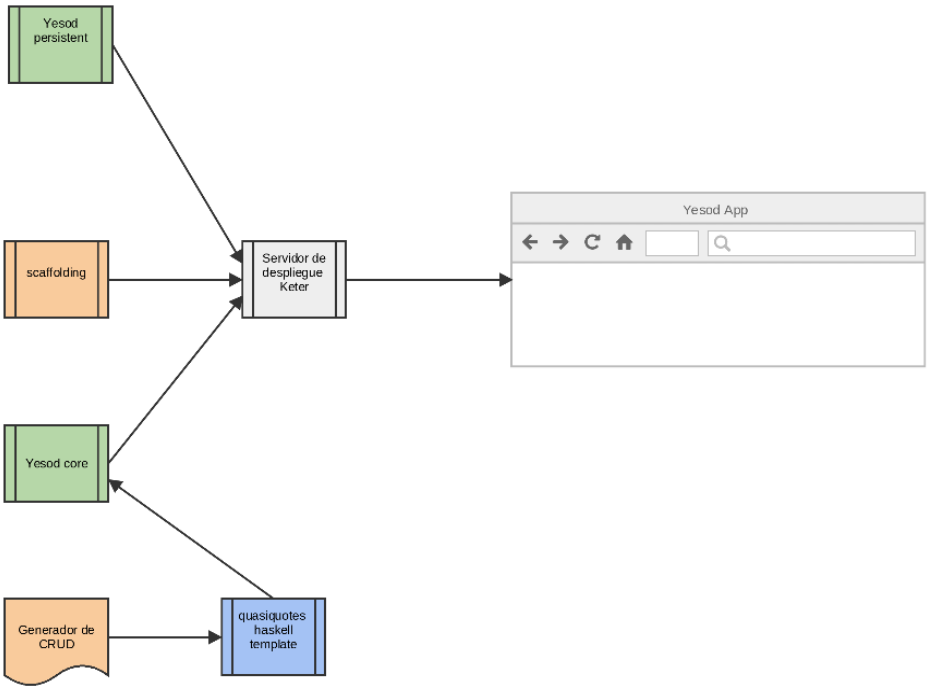
6.3 Presentación:

El siguiente gráfico es un diagrama ilustra la capa de presentación en el Framework Yesod.



6.4 Despliegue:

El siguiente gráfico es un diagrama ilustra el despliegue de la aplicación usando Keter.



PARTE III: Análisis y desarrollo del proyecto

A continuación se detallan las diferentes ideas que se tuvieron para el desarrollo del proyecto “Generación de operaciones de acceso de datos (CRUD) en función del modelo para el Framework de desarrollo web Yesod (Yesod-CRUD)”, los inconvenientes encontrados y sus soluciones.

1. Planteamientos preliminares: durante la etapa de definición, se tuvo como idea desarrollar un paquete que permitiera la automatización de las operaciones de CRUD para las aplicaciones desarrolladas en Yesod, en esta primera etapa se consideró deseable que el generador desarrollará los siguientes componentes

- Handlers de las operaciones.
- Interfaz para el uso de las 4 operaciones (crear, modificar, eliminar y listar).
- Formas.
- Rutas.

Sin embargo, debido a que el archivo de rutas se escribe en un DSL definido en Yesod en lugar de estar en Haskell, se descartó la generación de este archivo como una característica deseable. De hacerse, se generaría código no verificado y se presentan grandes dificultades a la hora de requerir eliminar o modificar alguna ruta. Sin embargo y por motivos de aprendizaje se desarrolló parcialmente esta funcionalidad, aunque no se incluyó dentro del paquete por los problemas antes mencionados.

Debido a las limitaciones de tiempo se decidió retirar del alcance de este proyecto la generación de formas, sin embargo dicha funcionalidad es altamente deseable y se incluye dentro del plan de continuidad.

2. Desarrollo del proyecto: El proyecto pasó por varias etapas y se realizaron varias aplicaciones en Yesod, que sirvieron de aprendizaje para el desarrollo del paquete. Estos fueron:

- **Generador de rutas**: utilizando las funciones para leer y escribir sobre archivos de Haskell se desarrolló una funcionalidad para generar el archivo de rutas. Dicho paquete se limitaba a escribir en un archivo las rutas para una entidad, sin embargo no permite modificar ni eliminar ya que el código generado se escribía en el archivo.
- **Ejemplo de CRUD**: Se creo un proyecto con una entidad de prueba que sirviera como referencia para las operaciones que debía generar el paquete.
- **Ejemplo de CRUD Subsitio**: Según el planteamiento inicial se consideró que las operaciones generadas deberías funcionar para SubSitios, es por ello que se desarrolló un proyecto de ejemplo con las cuatro operaciones dentro de un Subsitio.
- **Hamlet genérico**: Un elemento importante fue desarrollar interfaces genéricas para las operaciones. Éstas consisten de un Hamlet para crear y editar (La operación a realizar se define como parámetro de la función), un Hamlet para listar y finalmente un Julius para poder utilizar una operación HTML DELETE en lugar de POST para borrar.
- **Parser de operaciones algebraicas**: con el fin de aprender Template Haskell y QuasiQuote se desarrolló un parser de operaciones algebraicas usando ambos paquetes.

3. Métodos desarrollados: Este paquete buscar agilizar el desarrollo de aplicaciones usando Yesod sin limitar su uso; es por eso que se desarrollaron dos formas diferentes de generar las operaciones en una el programador debe codificar menos líneas de código, pero tiene grandes limitaciones para añadir nuevas funcionalidades en dichas operaciones; por el contrario con el segundo generador el usuario debe ingresar una mayor cantidad de líneas de código pero puede fácilmente añadir nuevas funcionalidades tales como autenticación, lógica de negocios, etc.

4. Métodos en contexto: una característica que tiene el generador de CRUD desarrollado es que los métodos no se escriben en ningún archivo, por el contrario se mantienen en algo que arbitrariamente e denominado “contexto de aplicación” pero que en la práctica hacer referencia a la forma como Template Haskell define el código generado; esta forma de generación facilita en gran medida su uso ya que uno puede seguir usando el generado para realizar cambios en la aplicación basta con re compilar, borrando, cambiando los parámetros o añadiendo una invocación.

5. Forma de generación de código: Las librerías que hicieron posible el desarrollo de este generador son Template Haskell y Quasiquotation; el modo de generación de código consiste en primer lugar en convertir los parámetros “String” recibidos en cada una de las funciones en funciones y tipos de datos de Haskell, allí se valida que el nombre de la entidad enviada por parámetro exista, luego de esto se genera un árbol de derivación sintáctica[10] con la función a generar incluyendo las funciones y tipos definidos a partir de los parámetros, una vez generado el árbol es compilado y de no presentarse errores las funciones son “generadas en el contexto de la aplicación” esto quiere decir que no se escriben en ningún archivo sino que están definidas y pueden ser usadas en el programa; pero cada vez que la aplicación es re compilada, las funciones vuelven a ser generadas. La gran ventaja de este generador en comparación de los que si escriben código en un archivo es que no se genera código si este no compila, el programador no necesita modificar código generado para implementar nuevas funcionalidades, tanto de lógica de negocio como de interfaz, y finalmente en caso de presentarse un cambio que implique añadir modificar o eliminar una de estas funciones generadas no se necesitará realizarlo a mano pasando archivo por archivo, bastará con hacer los cambios en la invocación del generador y recompilar la aplicación para que los cambios surtan efecto.

Los archivos desarrollados y que se entregan junto a este proyecto son:

- Proyecto de ejemplo con todas las operaciones de CRUD, llamado CRUD generic example.
- Proyecto de ejemplo con todas las operaciones de CRUD utilizando los dos generadores desarrollados llamado PGR-TH.
- Código fuente de la librería.
- Una maquina virtual con todas las configuraciones requeridas para poder correr los proyectos.

PARTE IV: Plan de continuidad

Basado en los resultados obtenidos en la elaboración del proyecto, se plantea el siguiente plan de continuidad:

1. **Desacoplar la librería del Scaffolding:** La librería actualmente requiere el contexto de la aplicación “App” para poder funcionar, sin embargo si se generalizan los tipos de las funciones generadas esto ya no sería necesario y se podría desacoplar la librería del proyecto.

2. **Autogeneración de formas:** Utilizando los mismas librerías de generación de código, es posible generar además la forma de cada entidad, para ello

se debe investigar a fondo la librería persistent, con el fin de poder obtener los atributos de una entidad (Entity)

3. **Soporte de formas con parámetro:** junto con el nombre de la entidad se debe permitir el envió de una lista de parámetros de la forma, esta funcionalidad tiene un impacto en la autogeneración de formas ya que se deben definir al momento de generar la forma

4. **Soporte de internacionalización:** Yesod tiene una funcionalidad de gran utilidad y es la internacionalización; actualmente el proyecto no soporta esta funcionalidad pero es posible incluirla dentro de la generación, para ello se deben tener en cuenta varias consideraciones, sin embargo mi recomendación es que los mensajes de la internacionalización deben ser enviados a los métodos, de nuevo el DSL puede ser la solución.

5. **DSL CRUD:** debido a que el usuario debe definir varias características del código a generar, se debería desarrollar un DSL que facilite y estandarice el uso de la librería.

6. **Subsistíos CRUD:** Brindar la posibilidad de que los métodos generados puedan ser usados tanto en el sitio como en un Subsistió.

Existe un abanico enorme de funcionalidades adicionales tales como parametrizar los estilos, generar la autenticación, generar el esquema de roles… entre otros, sin embargo los 6 elementos anteriores facilitan el uso de librería en ambientes de producción.

Parte V. Conclusiones y bibliografía

1. Conclusiones

- El uso de lenguajes de programación de alto nivel ya no ofrece una desventaja en términos de velocidad y eficiencia; por el contrario encontramos que el rendimiento de las aplicaciones escritos en lenguajes como Haskell son iguales o mayores a los ofrecidos por otros lenguajes de más bajo nivel
- En muchas aplicaciones, especialmente en aquellas que son web, el tiempo y costo que toma depurarlas puede llegar a ser igual o incluso mayor al de desarrollo; gracias a Haskell y Yesod, es posible desarrollar aplicaciones mucho más robustas con muy pocos errores en tiempo de ejecución y con ende con un costo muchísimo menor de depuración.

- La verificación de HTML y JavaScript es una importantísima ventaja a la hora de desarrollar aplicaciones en Yesod, ya que estos códigos generalmente presentan errores difíciles de encontrar y corregir.
- La gran mayoría de librerías que posee Yesod y en general Haskell son de código abierto; esto ofrece grandes ventajas a la hora de utilizarlas en los proyectos, y ha permitido que dichas librerías se desarrollen con gran velocidad.
- Actualmente se invierte mucho tiempo desarrollando las operaciones de CRUD de las aplicaciones; la programación de dichas operaciones, tanto la lógica como la interfaz, es repetitiva y es posible automatizarse
- La generación de código sobre un archivo ofrece facilidades tanto para desarrollarlos como para modificarlos, sin embargo unas vez el programador realiza un cambio a mano se hace casi obligatorio hacer los demás cambios a mano y se pierde la posibilidad de seguir usando el generador para hacer dichos cambios; además de no implementarse con cuidado se podría generar código errado difícil de corregir.

2. Bibliografía

1. Synfony [Consultado desde internet 21/04/2015] http://es.wikipedia.org/wiki/Symfony#Las_ventajas_tecnol.C3.B3gicas_de_Symfony
2. Synfony 1.0 la guía definitiva capítulo 14 generadores [Consultado desde internet 21/04/2015] http://librosweb.es/libro/symfony_1_0/capitulo_14.html
3. Introducción a Play! Framework [Consultado desde internet 21/04/2015] http://playdoces.appspot.com/documentation/1.2.4/overview
4. Play CRUD: generador de interfaces de administración [Consultado desde internet 21/04/2015] http://playdoces.appspot.com/documentation/1.2.3/crud
5. Yii Framework en español [Consultado desde internet 21/04/2015] http://yiiframeworkespanol.blogspot.com/2012/06/gii-generacion-automatica-de-codigo.html
6. Por qué Yesod: consultado desde internet el 07/06/2015 http://www.yesodweb.com/
7. Primeros pasos en Yesod: creado el 19/02/2015 https://docs.google.com/document/d/1BeUkhJBhhS98WvF_L1zwIXheABbKT7A69vcrs78lHAs/edit?usp=sharing
8. Mapeo Objeto relacional: Consultado desde internet el 07/06/2015 http://es.wikipedia.org/wiki/Mapeo_objeto-relacional
9. Persistent: consultado desde internet el 07/06/2015 http://www.yesodweb.com/book/persistent
10. Shakespearean Templates: consultado desde internet el 07/06/2015 http://www.yesodweb.com/book/shakespearean-templates
11. Widgets: consultado desde internet el 07/06/2015 http://www.yesodweb.com/book/widgets

12. Forms: consultado desde internet el 07/06/2015 http://www.yesodweb.com/book/forms
13. Routing and Handlers: consultado desde internet el 07/06/2015 http://www.yesodweb.com/book/routing-and-handlers
14. Template Haskell: consultado desde internet el 16/06/2015 https://wiki.haskell.org/Template_Haskell
15. Keter: consultado desde internet el 07/06/2015 https://github.com/snoyberg/keter