**HandBook - Manual Técnico**

Versión 1.0

22 de noviembre de 2016

Índice

[Introducción](#_2et92p0)

[Atributos no funcionales y de calidad](#_tyjcwt)

[Modelo arquitectónico](#_3dy6vkm)

[Modelo-vista-controlador](#_1t3h5sf)

[Tecnología y herramientas](#_4d34og8)

[Justificación de Arquitectura](#_2s8eyo1)

[Control de Versiones](#_lnxbz9)

[Diseño](#_1ksv4uv)

#

#

# Introducción

 Este manual de arquitectura pretende dar una descripción detallada de la arquitectura del sitio “Boarding Agents”. En sus contenidos podremos observar las características más importantes del sistema. En el handbook podremos encontrar los siguientes puntos:

* Atributos no funcionales y de calidad
	+ Requisitos que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.
* Modelo Arquitectónico
	+ Diseño de más alto nivel de la estructura de un sistema.
* Tecnologías y herramientas
	+ Se describen las herramientas y tecnologías empleadas en el proyecto.
* MVC
	+ Descripción del patrón de diseño que se utilizó en el sistema.
* Control de versiones
	+ Gestión de los diversos cambios que se realizan sobre los elementos del producto o una configuración del mismo.
* Despliegue
	+ Explicacion de como se realizó el despliegue del producto final.

# Atributos no funcionales y de calidad

Con base en las reuniones con Emprediem y los acuerdos tomados en ellas, se determinaron que los atributos de calidad esperados para el proyecto son los siguientes:

**Eficiencia**: El sistema debe ser capaz de soportar la información requerida para registrar 800 agentes de cambio.

**Accesibilidad**: El sistema debe ser accesible en dispositivos móviles por lo cual debe tener componentes responsivos.

**Portabilidad**: El sistema debe ser capaz de visualizarse en navegadores Chrome, Mozilla Firefox, Safari e Internet Explorer en las versiones del año 2015 en adelante.

**Seguridad**: Los agentes de cambio deben ser registrados a través de ligas únicas.

**Usabilidad**: La información de los agentes de cambio debe ser obtenida de su perfil de LinkedIn.

**Escalabilidad**: El sistema debe brindar la capacidad de crecimiento futuro acorde al modelo de negocio.

**Disponibilidad**: El sistema debe estar disponible para sus usuarios en todo momento

# Modelo arquitectónico

# Modelo-vista-controlador

Para la arquitectura de este proyecto, se eligió el modelo-vista-controlador por sus ventajas de forma de acomodar el código y porque los miembros del equipo ya han tenido experiencia previa con este patrón el cual consta de tres componentes principales: 

<https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Art/model_view_controller_2x.png>

* El *modelo*: es la parte que se encarga de guardar información.
* La *vista*: es la parte estética que se muestra al usuario de la aplicación.
* El *controlador*: contiene toda la funcionalidad que conecta a la vista con la información.

En otras palabras, el controlador es el que se encarga de la lógica detrás del vínculo de la información con la vista.

# Tecnología y herramientas

# Justificación de Arquitectura

Frameworks

Back-end

|  |  |  |  |
| --- | --- | --- | --- |
| Framework | Laravel | Ruby on rails | Symfony |
| Versión | 5.3 | 5.0.0.1 | 3.2 |
| Licencia | MIT | MIT | MIT |
| Requisitos | * PHP >= 5.6.4
* OpenSSL
* PDO
* Mbstring
* Tokenizer
* Composer
 | * Ruby >= 2.2.2
* RubyGems
 | * PHP >= 5.2
 |
| Arquitectura | MVC | MVC | MVC |
| Testing | PHPUnit | Unit testing | Tiene testing |
| Migración DB | Nativo | Nativo | Existe plugin |
| Autenticación | Nativa | Nativa | Nativa |
| Conocimiento del equipo | 5/7 | 2/7 | 0/7 |

Después de analizar la información recaudada se detectaron grandes similitudes entre las tres opciones ya que comparten o pueden tener las mismas funcionalidades. Sin embargo una de las opciones utiliza ruby, por lo cual para utilizar con comodidad y entendimiento el framework sería necesario aprender el lenguaje y posteriormente comenzar a practicar con el framework. Las dos opciones restantes utilizan PHP como base por lo cual se puede iniciar la capacitación en el framework sin aprender un nuevo lenguaje ya que todos los integrantes tienen conocimiento o han usado PHP. La diferencia entre los frameworks que utilizan PHP recae en que Symfony no tiene incluidos módulos que vienen de forma nativa en Laravel, además de que más de la mitad de los integrantes han utilizado Laravel mientras que Symfony es nuevo para todos.

Utilizando como referencia la experiencia del equipo y los requerimientos para iniciar la capacitación se tomó la decisión de utilizar Laravel como framework para el desarrollo del sistema.

La razón tras la elección de este framework en los anteriores proyectos de los integrantes parte de la presencia de un motor de plantillas llamado “*blade*”, que usa como fundamento la idea de escribir la menor cantidad de código repetido, modularizar, y aplicarlo en vistas. Por otra parte facilita trabajar con relaciones de base de datos por medio de su ORM (object-relational mapping) y a través de CLI (command line interface) poder crear o modificar archivos. Además de contar con una comunidad altamente activa para resolver dudas o problemas.

Front-end

|  |  |  |
| --- | --- | --- |
| Framework CSS | Bootstrap | Semantic UI |
| Versión | 3.3.7 | 2.2 |
| Licencia | MIT | MIT |
| Precompilador LESS | Incluido | Incluido |
| Precompilador SASS | Incluido | No incluido |
| Responsivo | Si | Si |
| Modular | Si | Si |
| Plantillas de inicio | Incluidas | No incluidas |
| Enfoque | Rapidez y movilidad | Semántica, ambivalencia de indicadores, y responsividad |

La selección para el front-end framework fue Semantic UI; inicialmente se tomó como opción principal Bootstrap pero tras comparar las estéticas Semantic UI tuvo una mejor aceptación en el equipo. Se eligió por tener una gran variación de elementos que pueden ser agregados al sistema y aportar una buena impresión estética. Además cuenta con una gran documentación muy fácil de entender, accesible y amigable; si surge algún problema, se puede recurrir a la comunidad que se encuentra en constante actividad.

# Control de Versiones

#

# Diseño

MER



Diagrama de despliegue

****

Diagramas Interacción - Comunicación

Proyectos



Historias de Cambio



Usuarios

Iniciar sesión



Cerrar sesión



Login LinkedIn



Logout LinkedIn



Modificar Usuarios



Inversiones de Capital Intelectual

Invertir Capital Intelectual



Aceptar rechazar o eliminar Inversion de capital intelectual



Chats

Crear chat



Canales de Comunicación



Aportación en canal de comunicación

