Proyecto

Questionfinder

**Historial de revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 22/03/2017 | 1.0 | Se realiza el diseño de la propuesta de negocios, el diseño del modelo entidad-relación y el diccionario de datos del proyecto. | José Ferrer |
| 29/03/2017 | 1.1 | Se verifica el diseño de la propuesta y se crea la lista de actividades del siguiente ciclo | José Ferrer |
| 13/04/2017 | 1.2 | Se añade la creación de modelos, controladores y rutas de la API a ofrecer | José Ferrer |
| 15/04/2017 | 1.3 | Se añade el procedimiento para cargar el código en repositorios y en servicios PaaS. | José Ferrer |
| 15/04/2017 | 1.4 | Se añaden las pruebas hechas a la API cargada en el servicio PaaS | José Ferrer |
| 16/04/17 | 1.5 | Se realizan revisiones finales al documento | Edgar Tarascio |

Contenido

[1.-Introducción 4](#_Toc480185504)

[1.1.- Propósito de este documento 4](#_Toc480185505)

[1.2.- Alcance 4](#_Toc480185506)

[2.- Descripción de metodología de trabajo 4](#_Toc480185507)

[2.1.- Fundamentos 4](#_Toc480185508)

[2.2.- Valores de trabajo 4](#_Toc480185509)

[3.- Distribución de Miembros y de Actividades 5](#_Toc480185510)

[3.1.- Personas involucradas y roles 5](#_Toc480185511)

[3.2.- Responsabilidades de los roles 5](#_Toc480185512)

[3.3.- Product Backlog 6](#_Toc480185513)

[4.- Artefactos (Primera iteración) 7](#_Toc480185514)

[4.1.- Sprint Backlog 7](#_Toc480185515)

[4.2.- Incremento 7](#_Toc480185516)

[4.3.- Cálculo de tiempos 15](#_Toc480185517)

[4.5.- Gráfico de Burn-down 16](#_Toc480185518)

[5.- Artefactos (Segunda iteración) 16](#_Toc480185519)

[5.1.- Sprint Backlog 16](#_Toc480185520)

[5.2.- Incremento 17](#_Toc480185521)

[5.3.- Cálculo de tiempos 20](#_Toc480185522)

[5.5.- Gráfico de Burn-down 22](#_Toc480185523)

[6.- Artefactos (Tercera iteración) 22](#_Toc480185524)

[6.1.- Sprint Backlog 22](#_Toc480185525)

[6.2.- Incremento 23](#_Toc480185526)

[6.3.- Cálculo de tiempos 27](#_Toc480185527)

[6.5.- Gráfico de Burn-down 29](#_Toc480185528)

[7.- SCRUM Retrospective 30](#_Toc480185529)

[7.1.- Cálculo de tiempos 30](#_Toc480185530)

[7.2.- Gráfico de Burn-up 31](#_Toc480185531)

[7.3.- Gráfico de Burn-down 31](#_Toc480185532)

# 1.-Introducción

En este documento se desarrolla la aplicación de la metodología SCRUM en el grupo Vertex Caster Studios para la gestión del desarrollo del proyecto Questionfinder. Se incluye el alcance del proyecto, la metodología de trabajo a usar, con sus valores y fundamentos, la distribución de roles entre las personas involucradas, la lista de elementos a desarrollar para el proyecto (Product Backlog), y, por cada iteración realizada, una lista de tareas realizadas, el incremento, los gráficos de burn-up y burn-down.

## 1.1.- Propósito de este documento

Documentar el desarrollo del proyecto en este archivo, en el cual se ilustre el incremento de características del proyecto Questionfinder a lo largo del tiempo.

## 1.2.- Alcance

Todas las personas y procedimientos relacionados con el proyecto Questionfinder.

# 2.- Descripción de metodología de trabajo

## 2.1.- Fundamentos

Las razones por las que se decidió utilizar la metodología iterativa e incremental SCRUM son las siguientes:

* La metodología utiliza un Sistema Modular que permite crear del proyecto Questionfinder una base mínima, e ir mejorando sus características a lo largo del tiempo.
* Lo anterior permite entregas frecuentes al cliente de las características en desarrollo del proyecto, poniendo en servicio características en poco tiempo para su posterior depuración.
* Debido a la naturaleza cambiante de las características del proyecto, se necesita una metodología que no se atasque en caso de que una de las características en progreso cambie o se elimine, o una nueva sea añadida.

## 2.2.- Valores de trabajo

Los valores de trabajo que se deben de practicar son los siguientes:

* Autonomía entre los miembros del equipo.
* Respeto en todo momento entre todos los miembros involucrados.
* Dedicación a las tareas
* Visibilidad y transparencia en la información

# 3.- Distribución de Miembros y de Actividades

## 3.1.- Personas involucradas y roles

Las personas involucradas y sus roles son:

|  |  |  |
| --- | --- | --- |
| **Persona** | **Contacto** | **Rol** |
| José Ferrer | jgfr2009@gmail.com / 0416-0777220 | Coordinador del equipo |
| José Urbina | urbinaj110@gmail.com / 0426-7291655 | Equipo Técnico / Documentador |
| Edgar Tarascio | elpanga1243@gmail.com / 0414-7110761 | Equipo Técnico / Desarrollador |
| José Abreu | joserabreuj@gmail.com /0424-7195688 | Equipo Técnico / Desarrollador |
| Jonhmer Bencomo | bjohnmer@gmail.com / 0424-7047809 | Gestor del producto |
|  |  |  |

## 3.2.- Responsabilidades de los roles

El Coordinador del equipo tiene las siguientes responsabilidades:

* Supervisar que las actividades asignadas para cada una de las iteraciones a realizar están siendo ejecutadas oportunamente.
* Comunicarse con el Product Owner en caso de deficiencias durante el curso de alguna de las iteraciones de desarrollo del proyecto.
* En caso de que el equipo presente problemas, discutir con todos los afectados las posibles soluciones que se pueden aplicar

El Documentador del equipo tiene las siguientes responsabilidades:

* Registrar cada uno de los cambios realizados en el proyecto.
* Documentar el uso de aquellas características que pudieran presentar algún problema para el trabajo del equipo, por desconocimiento de estas o por su mal uso.

El resto del equipo técnico tiene como responsabilidades lo siguiente:

* Desarrollar las tareas definidas en cada una de las iteraciones de desarrollo del proyecto.
* Tomar nuevas tareas en todo momento, priorizando aquellas que pertenecen a su capacidad.

El Gestor del Producto tiene las siguientes responsabilidades:

* Definir qué características se van a incluir para su planificación y desarrollo del producto en el backlog.
* Mantener registro de qué elementos del backlog se han desarrollado, que cambios hacer en los elementos existentes, o bien quitarlos o eliminarlos, durante las reuniones semanales.

## 3.3.- Product Backlog

El registro de elementos a desarrollar es el siguiente:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridad** |  | | **Nueva estimación en Sprint…** | | | |
| **Elemento** | **Estimación inicial (H)** | **1** | **2** | **3** | **SR** |
| 1 | Documentar las características comerciales y técnicas de la API a ofrecer. | 50 | 15 | 30 | 5 | - |
| 2 | Implementar el comportamiento básico de la API a ofrecer. | 120 | 50 | 50 | 20 | - |
| 3 | Preparar la API para su servicio en red. | 80 | 35 | 25 | 20 | - |
| 4 | Determinar e implementar el mecanismo de seguridad más apropiado para la API en desarrollo | - | - | - | - | - |
| 5 | Implementar un mecanismo para almacenar las imágenes como datos binarios en los registros de la API | - | - | - | - | - |
| 6 | Implementar mecanismos de filtrado y selección en los registros | - | - | - | - | - |
| 7 | Implementar un módulo de autenticación y registro de usuarios. | - | - | - | - | - |
| 8 | Implementar mecanismos para llevar la estadística de uso de la API | - | - | - | - | - |
| 9 | Diseñar y aplicar pruebas para probar la API en conjunto con sus nuevos módulos | - | - | - | - | - |
| 10 | Documentar el uso de los nuevos componentes creados de la API | - | - | - | - | - |
|  | Total de Horas de Desarrollo | 250 | 100 | 105 | 45 | - |

Se desglosan los elementos del backlog de la siguiente manera:

* El elemento “Documentar las características comerciales y técnicas de la API a ofrecer” consiste en desarrollar el modelo de negocio CANVAS del proyecto, desarrollar el modelo entidad relación y el diccionario de datos de cada una de las entidades a implementar.
* El elemento “Implementar el comportamiento básico de la API a ofrecer”, se implementarán las operaciones CRUD de cada una de las tablas que conforman el modelo ofrecido a través de la API, así como aquellas funcionalidades necesarias para el funcionamiento integral de esta.
* El elemento “Preparar la API para su servicio en red” consiste en preparar todos los detalles para ofrecer y respaldar la API en Internet, y así poder realizar las pruebas con los diferentes dispositivos objetivos de una forma más ubicua.
* El elemento “Determinar e implementar el mecanismo de seguridad más apropiado para la API en desarrollo” consiste en estudiar cuál de los diferentes mecanismos de seguridad que se pueden implementar en una API es el más conveniente e implementarlo, este mecanismo de seguridad incluye el hecho de programar el funcionamiento de la API de forma tal que no se exceda la cuota que el servicio PaaS usado no sobrepase las 24 horas al día.
* El elemento “Implementar un mecanismo para almacenar las imágenes como datos binarios en los registros de la API” consiste en buscar cómo hacer para que los datos recibidos de una imagen sean almacenados como datos binarios, de forma tal de que dichos datos puedan ser enviados a los clientes sin que estos tengan que acceder a una imagen en un servidor remoto.
* El elemento “Implementar mecanismos de filtrado y selección en los registros” consiste en implementar un mecanismo que filtre los datos cuando sus campos son nulos o bien si no se necesitan, y así optimizar el tamaño total de los datos enviados.
* El elemento “Implementar un módulo de autenticación y registro de usuarios” consiste en implementar un mecanismo que permita registrar el progreso de los usuarios a medida que resuelven cuestionarios.
* El elemento “Implementar mecanismos para llevar la estadística de uso de la API” consiste en implementar módulos de estadística para conocer el uso que los diferentes clientes hacen de las diferentes rutas implementadas en la API.
* El elemento “Diseñar y aplicar pruebas para probar la API en conjunto con sus nuevos módulos” consiste en diseñar diferentes pruebas para evaluar el funcionamiento de la API en desarrollo con los nuevos componentes implementados en el resto de las tareas.
* Y finalmente, el elemento “Documentar el uso de los nuevos componentes creados de la API” consiste en documentar cada uno de los nuevos componentes implementados en la API, y finalmente cargarla en el servicio PaaS elegido para servir dicha API.

# 4.- Artefactos (Primera iteración)

## 4.1.- Sprint Backlog

La lista de tareas a realizar es la siguiente, y se inicia el día 23 de marzo de 2017 y termina el día 30 de marzo de 2017:

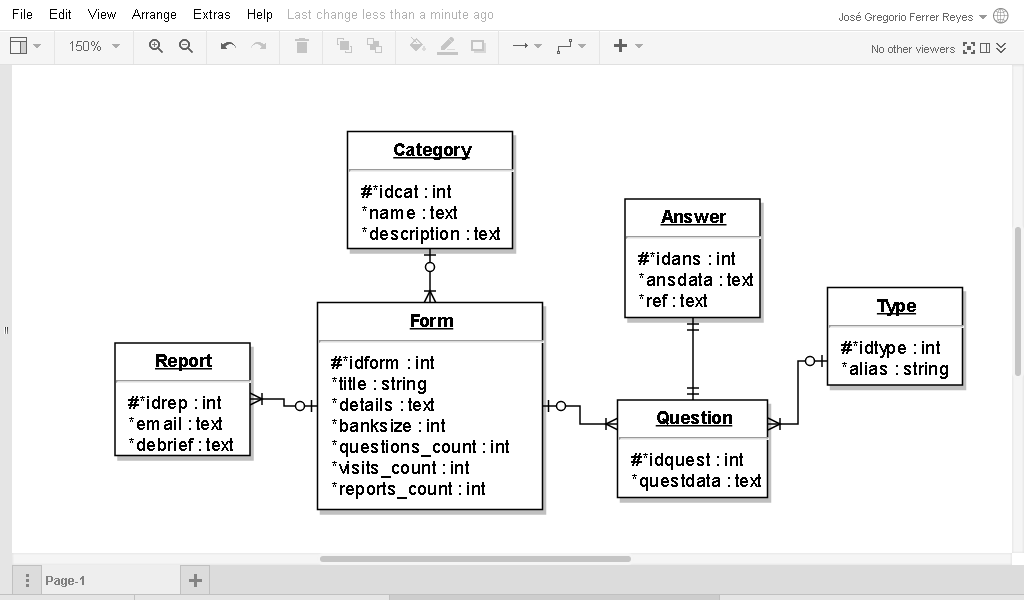
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | **Nueva estimación…** | | | | | | |
| **Elemento del backlog** | **Tareas** | **“Voluntario”** | **Estimación inicial (H)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| Documentar las características comerciales y técnicas de la API a ofrecer. | Desarrollar el modelo CANVAS del servicio que se va a implementar | José Ferrer | 5 | 3 | 2 | - | - | - | - | - |
| Diseñar el modelo entidad-relación del servicio a implementar | José Ferrer | 5 | 2 | 3 | - | - | - | - | - |
| Desarrollar el diccionario de datos para cada una de las entidades creadas | José Ferrer | 5 | 1 | 2 | 2 | - | - | - | - |
| Desarrollar la lista de los puntos finales del servicio a implementar | José Ferrer | 15 | 5 | 5 | 5 | - | - | - | - |
| Documentar el uso de la Aplicación Postman | José Abreu | 20 | 5 | 12 | 3 | - | - | - | - |
|  | Total | 50 | 16 | 24 | 10 | - | - | - | - |

## 4.2.- Incremento

El modelo CANVAS del servicio propuesto se encuentra en la página siguiente, y describe de forma detallada, cómo funciona el proyecto.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Fecha:** 21-3-17 | **Propuesta:** Questionfinder | **Empresa:** Vertex Caster Studios. | | | **Creadores:** José Ferrer, José Urbina, Ángelo Tarascio, Edgar Tarascio, Miguel Colmenares | **Iteración:** 1 |
| **8.- Socios Clave:**  - Usuarios quienes serán los que velarán por la calidad del contenido de la plataforma, y lo moderarán usando sus reportes.  - Librerías tanto físicas como digitales, las cuales ofrecerán las obras que tengan en existencia para su publicitación a los usuarios luego de resolver cuestionarios que tienen asociados diferentes bibliografías. | **6.- Actividades clave:**  - Mantenimiento y desarrollo de la plataforma web.  - Publicidad.  - Administración.  - Revisión de cuestionarios.  - Corrección de cuestionarios.  - Gestión de usuarios. | | | **1.- Propuesta de valor:**  Desarrollar una aplicación web adaptable que le permita a una persona acceder a diferentes cuestionarios relacionados con diversos temas, la persona puede ver una de últimos cuestionarios publicados o buscar una coincidencia de títulos, con la finalidad de aportar una herramienta más para el aprendizaje de muchos temas existentes. | **4.- Relación con los clientes:**  - Un usuario solicita la creación de un cuestionario dando el tema del cual quiere que se cree el cuestionario, o el conjunto de preguntas que quiere que se carguen.  - El moderador revisa que todo el contenido sea apto para la plataforma, y revisa aquellos cuestionarios que tengan muchos reportes, y carga los cuestionarios, verificando que respuestas se pueden asociar entre sí.  - Al cargar un cuestionario, el moderador informa al usuario que está listo, indicando el nombre del cuestionario creado.  - Los usuarios informan en caso de que los cuestionarios tengan errores. | **2.- Segmento de Mercado:**  -Estudiantes de educación media y superior, especialmente aquellos cuyos estudios los ponen en contacto con mucho conocimiento teórico.  -Profesores de educación media y superior, cuyas áreas contienen muchos conocimientos teóricos. |
| **7.- Recursos Clave:**  - Personal: Programador Web, Diseñador Web, Publicista, Docente especializado en Metodología del aprendizaje, Contador, Moderador, Gerente.  - Infraestructura: Plataforma Web, un local físico.  - Tecnología: Servidor, Dominio Web, Bases de datos, Telefonía.  - Capital: Computadores, Teléfonos, Equipo de Oficina y Bibliografía en Digital. | | | **3.- Canales:**  - Online, a través de la aplicación web adaptable a cualquier dispositivo que soporte el motor Webkit.  - Correo Electrónico, por el cual los usuarios solicitan la creación de los cuestionarios, el cual puede incluir un pago por publicación priorizada. |
| **9.- Estructura de costos:**  Sueldos y salarios, adquisición y mantenimiento de equipos tecnológicos, servicios públicos e impuestos, hosting de la plataforma, publicidad en línea, material bibliográfico en los casos que sea estrictamente necesario.  Modelo de negocio orientado al valor. | | | **5.- Fuente de Ingresos:**  - Promoción de cuestionarios: Los usuarios interesados por la rápida publicación de sus cuestionarios pueden pagar para que el o los moderadores publiquen antes sus cuestionarios.  - Publicidad de editoriales: Las tiendas que venden libros electrónicos y físicos podrán publicar obras que sean acordes a los cuestionarios que un usuario esté resolviendo.  Se aceptarán pagos vía transferencias y depósitos tanto para las publicaciones priorizadas como para la publicidad, que serán verificados antes de ser realizados. | | | |

El modelo entidad-relación del proyecto es el siguiente:



En este modelo entidad relación se tiene:

* Las categorías pueden tener relacionados arios cuestionarios.
* Los cuestionarios deben pertenecer a una categoría, pueden tener varios reportes, pueden tener varias preguntas.
* Los reportes deben tener asociado un cuestionario al cual hacen referencia.
* Cada pregunta debe de pertenecer a un cuestionario y a un tipo en particular, este tipo es lo que permite que el cliente pueda deducir como mezclar las posibles respuestas.
* Cada respuesta debe de responder a una sola pregunta.
* Un tipo en particular puede ser referido por muchas preguntas diferentes, así sean de diferentes cuestionarios.

El diccionario de datos para el proyecto es el siguiente:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Diccionario de datos** | | | | | |
| **Proyecto Questionfinder** | | | | | |
| **Entidad:** Answers | | | | | |
| **Descripción:** Esta entidad almacena las respuestas a cada una de las preguntas. | | | | | |
| **Atributo** | **Código del atributo** | **Nombre en la base de datos** | **Tipo de dato** | **Longitud** | **Descripción** |
| Identificador de la respuesta | idans | answer\_id | Entero largo | 1 | Este atributo almacena el identificador de cada respuesta |
| Respuesta almacenada | ansdata | ansdata | Texto | N/A | Este atributo almacena |
| **Operaciones afines** | | | | | |
| 1. \*Crear una respuesta, indicando también a que pregunta responde. 2. \*Eliminar una respuesta indicando su identificador. 3. \*Modificar una respuesta indicando su identificador. 4. \*Listar todas las respuestas de un cuestionario. 5. Cargar la respuesta de una pregunta dada. 6. Cargar las respuestas de un cuestionario.   \*Solo permitidas a los administradores del servicio. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Diccionario de datos** | | | | | |
| **Proyecto Questionfinder** | | | | | |
| **Entidad:** Category | | | | | |
| **Descripción:** Almacena las categorías de cada uno de los cuestionarios ofrecidos a los usuarios. | | | | | |
| **Atributo** | **Código del atributo** | **Nombre en la base de datos** | **Tipo de dato** | **Longitud** | **Descripción** |
| Identificador de categoría | idcat | category\_id | Entero  largo | 1 | Este atributo identifica de forma univoca cada uno de las categorías donde se agrupan los cuestionarios |
| Nombre de la categoría | name | name | Texto | N/A | Este atributo indica el nombre dado a cada categoría almacenada |
| Descripción de la categoría | description | description | Texto | N/A | Este atributo sirve para describir que temas pueden entrar en cada una de las categorías almacenadas |
| **Operaciones afines** | | | | | |
| 1. \*Crear una categoría dado el nombre y la descripción de esta. 2. \*Eliminar una categoría dado su identificador. 3. \*Modificar el nombre y/o la descripción de una categoría. 4. \*Consultar los datos de una categoría en particular. 5. \*Listar todas las categorías almacenadas. 6. Dado el identificador de una categoría, cargar la información de cada uno de los cuestionarios asociados.   \*Solo permitidas a los administradores del servicio. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Diccionario de datos** | | | | | |
| **Proyecto Questionfinder** | | | | | |
| **Entidad:** Form | | | | | |
| **Descripción:** Almacena la información general de cada uno de los cuestionarios que se pueden resolver en el servicio que se piensa ofrecer a los potenciales usuarios. | | | | | |
| **Atributo** | **Código del atributo** | **Nombre en la base de datos** | **Tipo de dato** | **Longitud** | **Descripción** |
| Identificador de cuestionario | idform | form\_id | Entero largo | 1 | Este atributo se usa para identificar de forma univoca a cada uno de los cuestionarios almacenados |
| Título del cuestionario | title | title | Texto | N/A | Este atributo almacena el nombre de cada cuestionario almacenado. |
| Detalles del cuestionario | details | details | Texto | N/A | Este atributo almacena una descripción acerca del cuestionario, como una forma de permitirles a los usuarios saber de forma general qué se pregunta. |
| Cantidad de preguntas a realizar | banksize | banksize | Entero | 1 | Este atributo indica cuantas preguntas se van a tomar para ejecutar el cuestionario. |
| Cuenta de preguntas almacenadas | questions\_count | questions\_count | Entero | 1 | Este atributo lleva la cuenta de cuantas preguntas tiene un cuestionario. |
| Cuenta de visitas | visits\_count | visits\_count | Entero | 1 | Este atributo almacena cuantas veces se ha pedido la carga de las preguntas y sus respuestas |
| Cuenta de reportes | reports\_count | reports\_count | Entero | 1 | Aquí se indican cuantos reportes de errores del formulario se han hecho. |
| **Operaciones afines** | | | | | |
| 1. \*Crear un nuevo cuestionario. 2. \*Eliminar un cuestionario dado su identificador. 3. \*Modificar los datos de un cuestionario dado su identificador 4. Consultar la información de un cuestionario. 5. Listar todos los cuestionarios por fecha desde el más reciente. 6. Dado el identificador de un cuestionario, cargar sus preguntas, las respuestas asociadas a estas, e incrementar el contador de visitas al cuestionario. 7. \*Dado el identificador de un cuestionario, cargar todos los reportes realizados.   \*Solo permitidas a los administradores del servicio. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Diccionario de datos** | | | | | |
| **Proyecto Questionfinder** | | | | | |
| **Entidad:** Questions | | | | | |
| **Descripción:** Esta entidad almacena las preguntas de cada cuestionario. | | | | | |
| **Atributo** | **Código del atributo** | **Nombre en la base de datos** | **Tipo de dato** | **Longitud** | **Descripción** |
| Identificador de la pregunta | idquest | question\_id | Entero largo | 1 | Este atributo almacena el identificador único de cada pregunta almacenada. |
| Datos de la pregunta | questdata | questdata | Texto | N/A | Este atributo almacena la pregunta en forma textual. |
| **Operaciones afines** | | | | | |
| 1. \*Crear una pregunta, indicando también a que pregunta responde. 2. \*Eliminar una pregunta indicando su identificador. 3. \*Modificar una pregunta indicando su identificador. 4. \*Listar todas las pregunta de un cuestionario. 5. Cargar una pregunta dado su identificador. 6. Cargar las preguntas de un cuestionario, indicando el identificador del último.   \*Solo permitidas a los administradores del servicio. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Diccionario de datos** | | | | | |
| **Proyecto Questionfinder** | | | | | |
| **Entidad:** Reports | | | | | |
| **Descripción:** Almacena la información general de los reportes que los usuarios llenan cada vez que consiguen un error en el cuestionario, luego de verificarlo. | | | | | |
| **Atributo** | **Código del atributo** | **Nombre en la base de datos** | **Tipo de dato** | **Longitud** | **Descripción** |
| Identificador del reporte | idreport | report\_id | Entero largo | 1 | Este atributo se usa para identificar de forma única cada uno de los reportes almacenados. |
| Datos del reporte | debrief | debrief | Texto | N/A | Aquí se almacena los comentarios y mejoras propuestas por los usuarios |
| **Operaciones afines** | | | | | |
| 1. Crear un reporte, indicando el identificador del cuestionario al cual se le quiere crear. 2. \*Eliminar un reporte dado su identificador. 3. \*Modificar la información de un reporte, indicando su identificador. 4. \*Listar todos los reportes de un cuestionario en particular. 5. \*Mostrar los datos de un reporte en particular, indicando a qué cuestionario referencia.   \*Solo permitidas a los administradores del servicio. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Diccionario de datos** | | | | | |
| **Proyecto Questionfinder** | | | | | |
| **Entidad:** Types | | | | | |
| **Descripción:** Esta entidad permite relacionar las preguntas, para que cuando se generen los formularios, el cliente sea capaz de tomar respuestas que tengan un parecido en su estructura. | | | | | |
| **Atributo** | **Código del atributo** | **Nombre en la base de datos** | **Tipo de dato** | **Longitud** | **Descripción** |
| Identificador del tipo | idtype | type\_id | Entero largo | 1 | Este atributo almacena el identificador único de cada tipo disponible. |
| Alias del tipo | alias | alias | Texto | N/A | Aquí se almacena el nombre legible por humanos del tipo |
| **Operaciones afines** | | | | | |
| 1. \*Crear un nuevo tipo. 2. \*Modificar el alias de un tipo. 3. \*Eliminar un tipo indicando su número identificador. 4. \*Listar los tipos disponibles. 5. \*Consultar el alias de un tipo y su fecha de creación.   \*Solo permitidas a los administradores del servicio. | | | | | |

El siguiente es el listado de puntos de acceso, y la operación que permiten, de cara a los usuarios:

|  |  |  |  |
| --- | --- | --- | --- |
| **Verbo** | **Punto de acceso** | **Tabla** | **Operación que implementa** |
| GET | /v1/categories/ | Category | Listar todas las categorías |
| GET | /v1/categories/:id | Category | Mostrar los detalles de una categoría en particular |
| POST | /v1/categories/:id | Category | Crear una nueva categoría |
| PUT/PATCH | /v1/categories/:id | Category | Modificar una categoría existente |
| DELETE | /v1/categories/:id | Category | Eliminar una categoría existente |
| GET | /v1/categories/:id/getforms | Category→Forms | Cargar los cuestionarios dentro de una categoría en particular |
| GET | /v1/forms/ | Forms | Listar todos los formularios |
| GET | /v1/forms/:id | Forms | Cargar un solo formulario |
| POST | /v1/forms/:id | Forms | Crear un nuevo formulario |
| PUT/PATCH | /v1/forms/:id | Forms | Modificar un formulario existente |
| DELETE | /v1/forms/:id | Forms | Eliminar un formulario existente |
| GET | /v1/forms/:id/getdata | Forms→  (Questions+Answers) | Cargar las preguntas con sus respectivas respuestas de un formulario existente |
| GET | /v1/forms?term=:term | Forms | Buscar en el título y en los detalles de los formularios almacenados los términos indicados |
| GET | /v1/forms/:id/getreports | Forms→Reports | Cargar los reportes almacenados para un formulario en particular |
| GET | /v1/questions/ | Questions | Cargar todas las preguntas almacenadas en la base de datos |
| GET | /v1/questions/:id | Questions | Cargar una sola pregunta dada |
| POST | /v1/questions/:id | Questions | Crear una nueva pregunta |
| PUT/PATCH | /v1/questions/:id | Questions | Modificar una pregunta existente |
| DELETE | /v1/questions/:id | Questions | Eliminar una pregunta existente con su respectiva respuesta |
| GET | /v1/answers/ | Answers | Listar todas las respuestas almacenadas |
| GET | /v1/answers/:id | Answers | Cargar una sola respuesta |
| POST | /v1/answers/:id | Answers | Crear una nueva respuesta |
| PUT/PATCH | /v1/answers/:id | Answers | Modificar una respuesta existente |
| DELETE | /v1/answers/:id | Answers | Eliminar una respuesta de la base de datos |
| GET | /v1/types/ | Types | Cargar todos los tipos de preguntas almacenados en la base de datos |
| GET | /v1/types/:id | Types | Cargar un tipo en particular |
| POST | /v1/types/:id | Types | Crear un nuevo tipo de pregunta |
| PUT/PATCH | /v1/types/:id | Types | Modificar el alias de un tipo de pregunta |
| DELETE | /v1/types/:id | Types | Eliminar un tipo de pregunta existente |
| GET | /v1/reports/ | Reports | Cargar todos los reportes almacenados |
| GET | /v1/reports/:id | Reports | Cargar un solo reporte, según su índice |
| POST | /v1/reports/:id | Reports | Crear un nuevo reporte |
| PUT/PATCH | /v1/reports/:id | Reports | Modificar un reporte existente |
| DELETE | /v1/reports/:id | Reports | Eliminar un reporte de la base de datos |
|  |  |  |  |

## 4.3.- Cálculo de tiempos

El tiempo invertido para cada una de las actividades desarrolladas fue el siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elementos del backlog | Tareas | Tiempo estimado inicial (H) | Tiempo utilizado (H) | Diferencia |
| Documentar las características comerciales y técnicas de la API a ofrecer | Desarrollar el modelo CANVAS del servicio que se va a ofrecer | 5 | 3 | -2 |
| Diseñar el modelo entidad-relación del servicio a implementar | 5 | 4 | -1 |
| Desarrollar el diccionario de datos para cada una de las entidades creadas | 5 | 4 | -1 |
| Desarrollar la lista de los puntos finales del servicio a implementar | 15 | 15 | 0 |
| Documentar el uso de la aplicación Postman | 20 | 18 | -2 |
| Total | 50 | 44 | -6 |

La diferencia se justifica por las siguientes razones:

1. El diagrama del modelo ER ya estaba elaborado antes de empezar la iteración, por lo que solo fue un trabajo de limpieza.
2. La lista de puntos finales de la API del proyecto también esta lista parcialmente antes de iniciar el proyecto, por lo que no se destinó tiempo completo a su desarrollo.
3. Los diccionarios de datos tienen más tiempo del estimado porque se han tenido que revisar múltiples veces.

**4.4.- Gráfico de Burn-up**

## 4.5.- Gráfico de Burn-down

# 

# 5.- Artefactos (Segunda iteración)

## 5.1.- Sprint Backlog

La lista de tareas a realizar es la siguiente, y se inicia el día 23 de marzo de 2017 y termina el día 30 de marzo de 2017

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | Nueva Estimación | | | | | | |
| **Elemento del backlog** | **Tareas** | **“Voluntario”** | **Estimación inicial (H)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| Implementar el comportamiento básico de la API a ofrecer | Crear los modelos base de la API que se va a ofrecer a los usuarios | José Ferrer | 20 | 10 | 5 | 5 | - | - | - | - |
| Implementar las rutas creadas en la lista de puntos de acceso | José Ferrer | 15 | 7 | 8 | - | - | - | - | - |
| Implementar las relaciones entre los modelos creados de la API | José Ferrer | 30 | 12 | 15 | 3 | - | - | - | - |
| Documentar la creación de rutas personalizadas con Ruby on Rails | José Abreu | 18 | 9 | 2 | 7 | - | - | - | - |
| Crear los controladores y rutas que ofrecerán las funciones personalizadas | José Ferrer | 17 | 10 | 3 | 4 | - | - | - | - |
| Rellenar con datos de prueba los diferentes modelos creados | José Ferrer | 20 | 2 | 15 | 3 | - | - | - | - |
|  | **TOTAL** | 120 | 50 | 48 | 22 | - | - | - | - |

## 

## 5.2.- Incremento

Los modelos de las entidades presentes en la API a desarrollar se crearon usando el siguiente script de bash:

*rails d scaffold v1/Answer*

*rails d scaffold v1/Question*

*rails d scaffold v1/Report*

*rails d scaffold v1/Form*

*rails d scaffold v1/Type*

*rails d scaffold v1/Category*

*rails db:drop*

*rails g scaffold v1/Category name:text description:text*

*rails g scaffold v1/Type alias:string*

*rails g scaffold v1/Form title:text details:text banksize:integer questions\_count:integer visits\_count:integer reports\_count:integer v1\_category:references*

*rails g scaffold v1/Report email:string debrief:text v1\_form:references*

*rails g scaffold v1/Question questdata:text v1\_form:references v1\_type:references*

*rails g scaffold v1/Answer ansdata:text ref:text v1\_question:references*

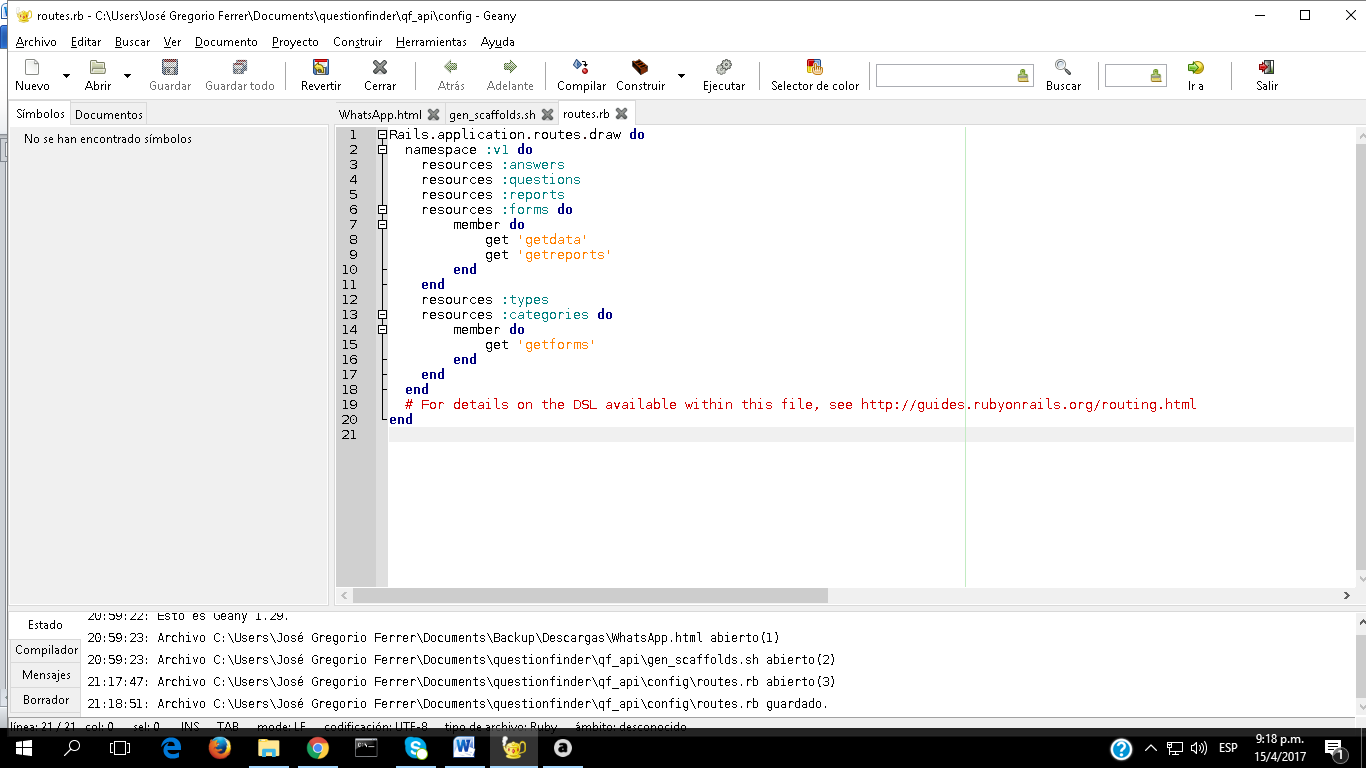
*rails db:create*

*rails db:migrate*

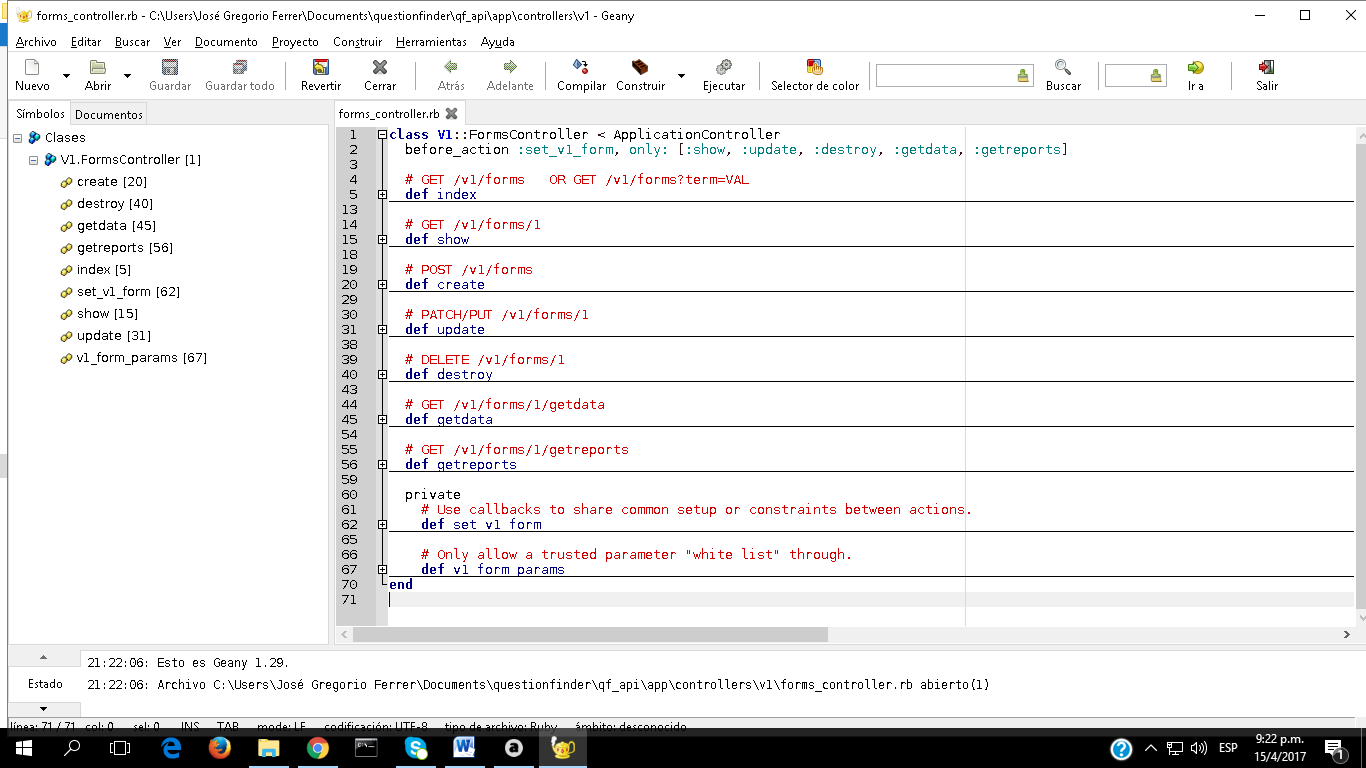
Básicamente, lo que hace este script es destruir todos los scaffolds creados previamente cuyos nombres coincidan con los de las tablas, en orden ascendente (es decir, desde los hijos, a los padres), luego, destruye la base de datos antigua, luego crea de nuevo los scaffolds con los mismos nombres anteriores, pero con los nuevos campos, y finalmente crea y migra la base de datos del proyecto de acuerdo a la información de los modelos recién creados.

Con el mismo script mostrado arriba, la mayoría de las rutas son creadas, por lo que no es necesario más esfuerzo del aplicado para crear los modelos.

Para crear rutas personalizadas, se debe de determinar si la operación aplica sobre toda una tabla, o sobre un registro específico de esta, si depende de un registro, es una ruta miembro (usando la palabra clave *member*), sino, si la operación necesita toda una tabla, es una colección (la cual se crea con *collection*), la cual se añade a la lista de operaciones que manipulan los recursos usando una estructura do-end luego del operador de recursos, luego, dentro de ese do-end, se usa la palabra clave correspondiente, seguido de otro do-end, y dentro de este, las rutas indicando su verbo y la ruta con la que coincide el llamado. En otras palabras, es así:



Luego de crear las rutas, se debe de editar la segunda línea en el controlador respectivo, con la finalidad de dar permiso de acceso al método que llama la ruta personalizada, editando el arreglo *:only []*, luego de ello se añade la o las funciones a la que se le dio permiso dentro de las funciones públicas del controlador en cuestión, así:



En caso de querer parámetros personalizados (como es el caso de la función index), se usa el objeto *params*, el cual indica que valores son obligatorios de pasar a una ruta (con el método *.require()*), y que valores se pueden pasar opcionalmente (con el método *.permit()*), se añade el valor de la variable personalizada, y se recupera de la ruta usando el objeto *params[:variablep]*, donde variablep es la variable personalizada que se quiere pasar a la función miembro del controlador.

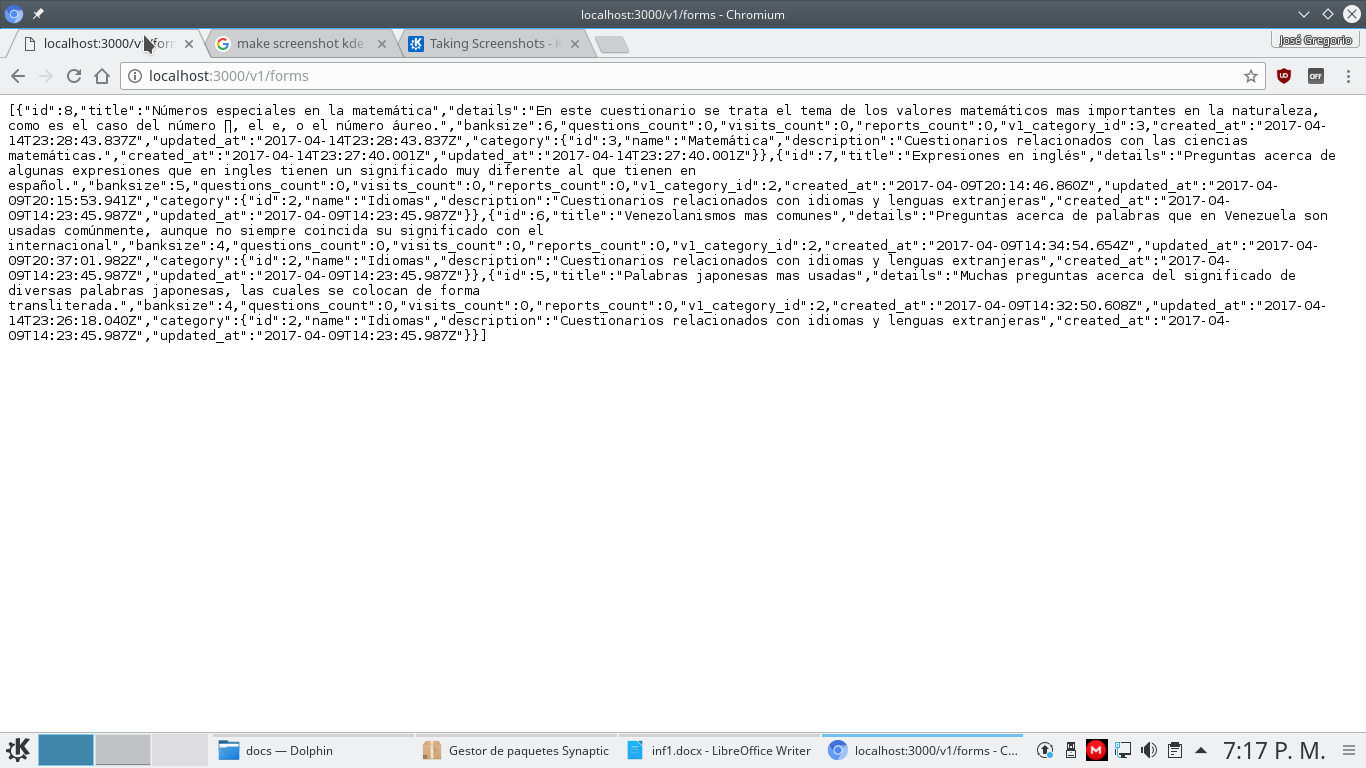
Apoyándose en lo anterior, se crearon 4 rutas personalizadas:

* Una ruta en la cual, dado el identificador de una categoría, se carguen solo los cuestionarios pertenecientes a dicha categoría.
* Una ruta en la cual, dado el identificador de un formulario, se carguen solo sus preguntas y respuestas.
* Una ruta en la cual, dado el identificador de un formulario, se carguen solo los reportes relacionados con dicho cuestionario.
* Finalmente, una ruta variable en la función index, en la cual, si no se pasa una palabra de búsqueda, se cargan todos los cuestionarios, sino, se cargan solo los cuestionarios en las cuales coincida el titulo o la descripción con alguna parte de la palabra clave.

Las tres primeras rutas son sencillas de implementar: Solo es necesario indicar el hijo de los padres que se quiere consultar, y, en el caso de las preguntas y respuestas, se usaron dos arreglos, en una va la pregunta, y en el otro, y en la misma posición, la respuesta (es decir, la respuesta 1 va con la pregunta 1, de forma que un índice encuentra el par con un mismo índice).

La cuarta ruta, por otro lado, requiere un poco más de lógica, porque, si hay un término de búsqueda, se debe llamar a una función de búsqueda creada en el modelo (la cual se encarga de filtrar según el parámetro pasado por la ruta), y si no lo hay, se deben de cargar todos los formularios existentes en la base de datos, incluyendo la categoría a la que pertenecen.

Con el relleno de datos, se probó que la API funciona bien si es utilizada en un servidor local, para ello, se crearon 4 registros diferentes:



## 5.3.- Cálculo de tiempos

El tiempo invertido para cada una de las actividades desarrolladas fue el siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del backlog** | **Tareas** | **Estimación inicial (H)** | **Tiempo utilizado (H)** | **Diferencia** |
| Implementar el comportamiento básico de la API a ofrecer | Crear los modelos base de la API que se va a ofrecer a los usuarios | 20 | 18 | -2 |
| Implementar las rutas creadas en la lista de puntos de acceso | 15 | 13 | -2 |
| Implementar las relaciones entre los modelos creados de la API | 30 | 33 | +3 |
| Documentar la creación de rutas personalizadas con Ruby on Rails | 18 | 20 | +2 |
| Crear los controladores y rutas que ofrecerán las funciones personalizadas | 17 | 17 | 0 |
| Rellenar con datos de prueba los diferentes modelos creados | 20 | 20 | 0 |
| Total | 120 | 121 | +1 |

La diferencia se justifica por las siguientes razones:

1. El método de utilizar scaffolds para crear los controladores y los modelos, en conjunto con el uso del script presentado arriba, aceleraron la implementación del esqueleto de la API en desarrollo.
2. La documentación de las rutas personalizadas se tomó tiempo porque se debieron realizar algunas pruebas antes de implementar el código definitivo en la API.
3. La implementación de las relaciones se tomó más tiempo del previsto, porque no solo se incluyen las relaciones en sí, sino las funciones callback que se ejecutan cuando se aplican las restricciones que las relaciones definen.

**5.4.- Gráfico de Burn-up**

## 5.5.- Gráfico de Burn-down

# 6.- Artefactos (Tercera iteración)

## 6.1.- Sprint Backlog

La lista de tareas a realizar es la siguiente, y se inicia el día 23 de marzo de 2017 y termina el día 30 de marzo de 2017:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | **Nueva Estimacion** | | | | | | |
| **Elemento del backlog** | **Tareas** | **“Voluntario”** | **Estimación inicial (H)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| Preparar la API para su servicio en red | Crear el repositorio de GitHub con el código del proyecto | José Ferrer | 5 | 5 | - | - | - | - | - | - |
| Configurar y cargar el repositorio  local | José Ferrer | 5 | 2 | 3 | - | - | - | - | - |
| Crear la cuenta  en Heroku | José Ferrer | 5 | 2 | 2 | 1 | - | - | - | - |
| Configurar el servicio para  alojar la API | José Ferrer | 20 | 8 | 2 | 5 | 4 | 1 | - | - |
| Cargar el código de la API a  Heroku | José Ferrer | 25 | 5 | 5 | 5 | 5 | 5 | - | - |
| Probar con datos la API remota | José Urbina | 5 | 1 | 2 | 2 | - | - | - | - |
| Documentar  el desarrollo | Edgar Tarascio | 15 | 5 | 5 | 5 | - | - | - | - |
|  |  | **Total** | 80 | 28 | 19 | 18 | 9 | 6 | - | - |
|  |  |  |  |  |  |  |  |  |  |  |

## 

## 6.2.- Incremento

El repositorio utilizado es el siguiente:

**https://github.com/dnuke/questionfinder-api/**  
  
Para configurar el repositorio local se ejecutan los siguientes pasos:

1. Antes que todo, se debe de configurar una clave RSA con la cual se le permitirá a la computadora cargar el código a Git, esto se hace de la siguiente forma:
   1. Se debe de crear una clave público-privada usando *ssh-keygen -t rsa -b 4096 -C "[correo electrónico]"*, donde “correo electrónico” es la dirección de correo del usuario, esto va a pedir dos veces una contraseña, la cual debe de ser la misma en los dos intentos, la contraseña queda a elección del usuario que configura la cuenta.
   2. Luego, se procede a agregar la clave creada al repositorio de claves, esto se hace ejecutando primero *eval "$(ssh-agent -s)"*, y luego se ejecuta *ssh-add ~/.ssh/id\_rsa*, donde id\_rsa es el archivo de claves creado.
   3. Luego se copia esa clave usando un programa llamado xclip, usando el siguiente comando: *xclip -sel clip < ~/.ssh/id\_rsa.pub*, si no se tiene xclip se puede instalar haciendo *apt-get install xclip.*
   4. En la cuenta de GitHub creada, se debe de agregar la nueva clave, la cual se añade accediendo a la configuración del usuario (este botón se encuentra en la esquina superior derecha de la ventana de GitHub), y en la configuración (Settings), se debe de seleccionar la opción SSH and GPG Keys, en la cual se añade la nueva clave, en conjunto con un nombre que describa en qué computadora se está utilizando dicha clave.
2. Luego se debe acceder a la carpeta que contiene el repositorio usando el comando *cd*.
3. Luego, se debe de inicializar el directorio como un repositorio de Git, esto se hace usando *git init.*
4. Luego, se hace *git add .*, con ello se incluyen todos los archivos que se quieren cargar en el repositorio.
5. Luego de añadir los archivos que se quieren cargar para el commit, se debe de indicar el nombre del commit, esto se hace con git commit -m “[descripción]”, donde descripción es un resumen de los cambios realizados en el commit
6. Luego de cargar la descripción del commit, se hace, git push origin master, para cargar desde el origen (el computador local) al maestro (el repositorio de GitHub).

Para cargar una aplicación (no importa el framework o el lenguaje) en Heroku, primero se debe de crear una cuenta de usuario, esta empresa ofrece tres planes, uno gratuito que puede funcionar 18 horas al día, uno para consumidor pequeño que ofrece tiempo completo, y otro plan para empresas que se caracteriza por su escalabilidad, usando el plan gratuito, se procede luego a instalar la utilidad de Heroku, esto se hace siguiendo los siguientes pasos:

1. Agregue el repositorio del kit de herramientas de Heroku con el siguiente comando:   
   *add-apt-repository "deb https://cli-assets.heroku.com/branches/stable/apt ./"*
2. Luego cargue la clave GPG del repositorio de Heroku con: *curl -L https://cli-assets.heroku.com/apt/release.key | sudo apt-key add -*
3. Actualice los repositorios con un *apt-get update*, y luego instale el paquete  
   con *apt-get install heroku*

Luego de instalado Heroku, inicie sesión con la misma cuenta con la que se identificó para poder cargar el código de git, usando: *heroku login*.

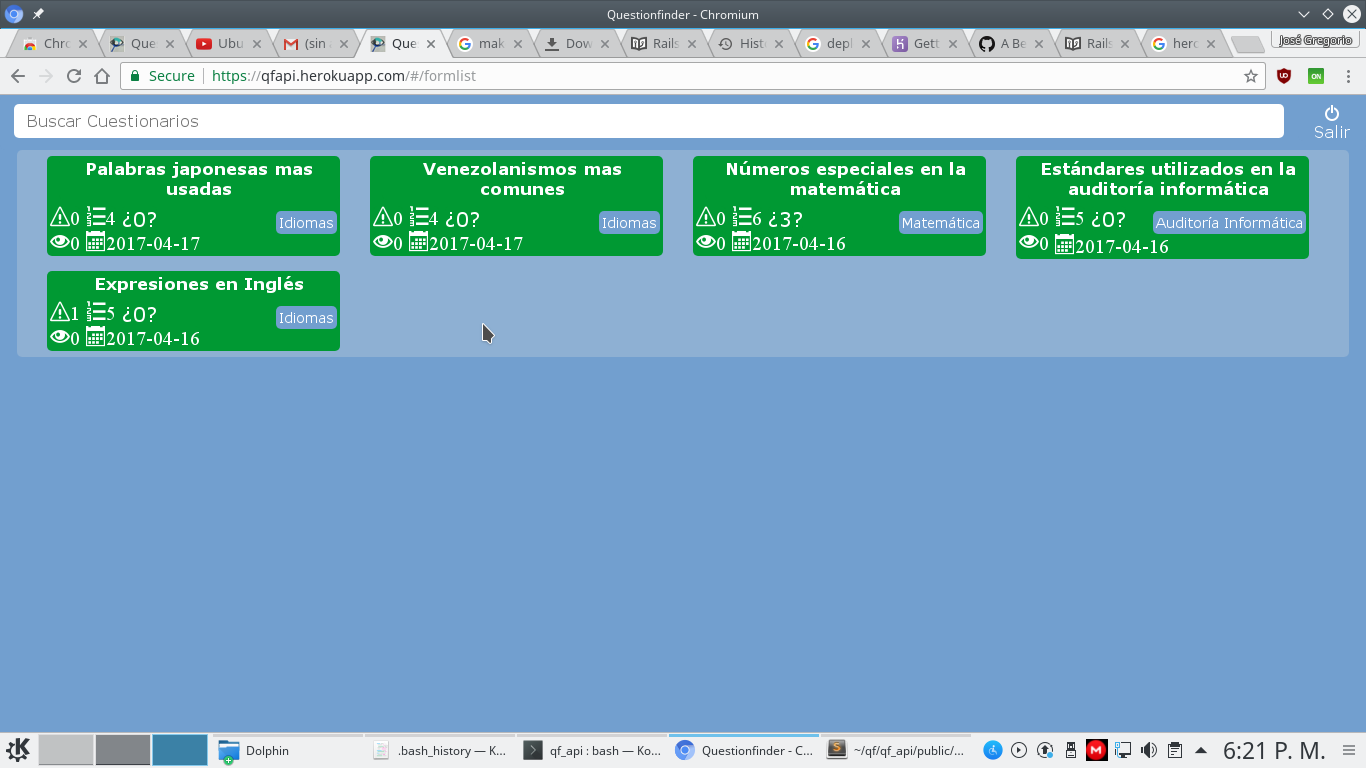
Con el comando anterior, el sistema le preguntara por el usuario y la contraseña usada para registrase en el servicio de Heroku, luego se utiliza *heroku create [nombre de la aplicación a cargar]* para crear un nuevo contenedor en el servicio (en el proyecto en particular, se usó qfapi, siendo la dirección que aloja la API desarrollada qfapi.herokuapp.com).

Para cargar el código del proyecto, es necesario estar en la carpeta donde se encuentra el código fuente del proyecto, ejecutar *heroku git:remote -a qfapi*, y luego, como si de una rama se tratase, se ejecuta: *git push heroku master*, esto carga el código al servidor remoto creado, luego de cargar el código, es necesario correr *heroku run rails db:create, luego heroku run db:migrate*  para crear la base de datos y migrar los modelos contenidos dentro de la aplicación.

[Probar con datos la API remota, 15H]

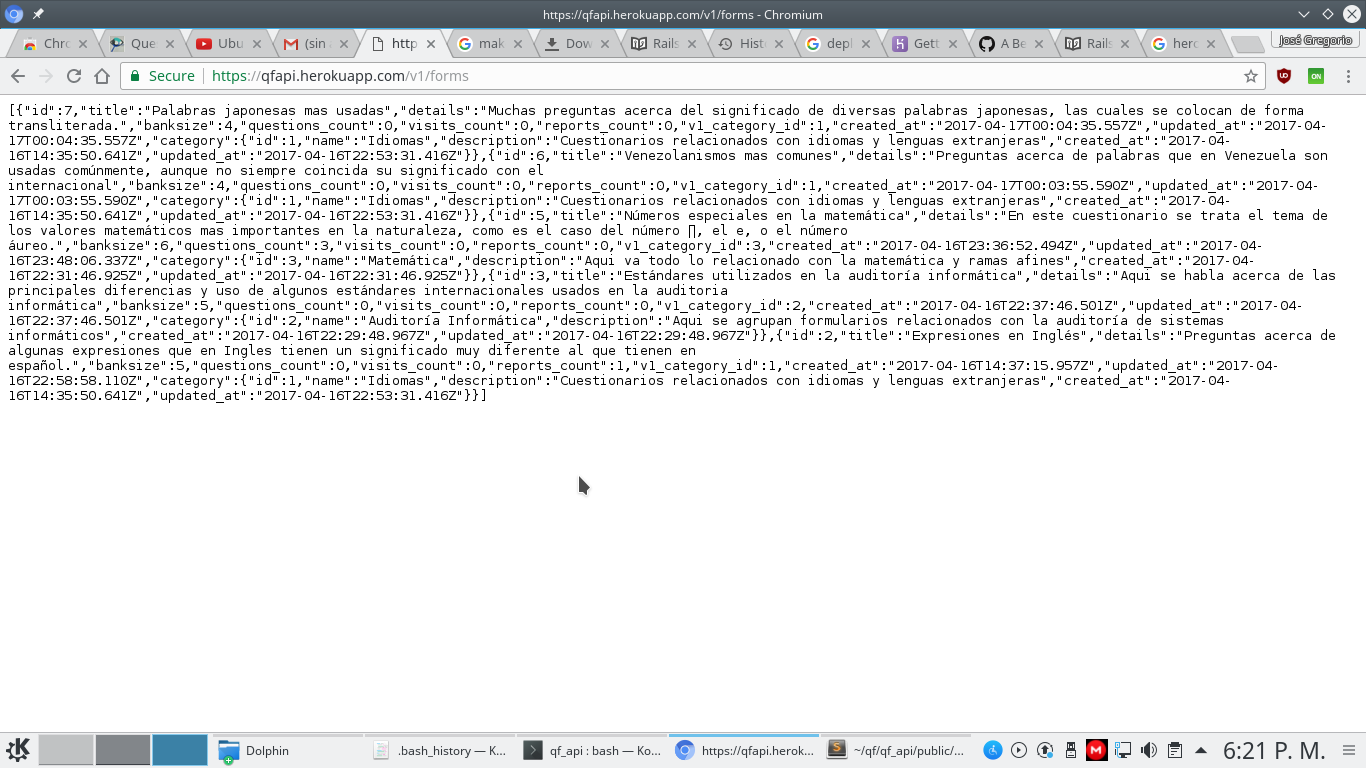
Luego de cargar el proyecto, es posible utilizar una aplicación que utilice la API creada, es más, se puede usar el mismo sistema de la API para poder contenerla, ya que en Rails el directorio public se comporta básicamente como una ruta directa que puede contener aplicaciones, por lo que es posible contener tanto una API, como una aplicación de tipo SPA que hagan uso de dicha API.

Como por ejemplo, si se llama la ruta directa del proyecto, se ejecuta la aplicación que esta enlazada con la API:

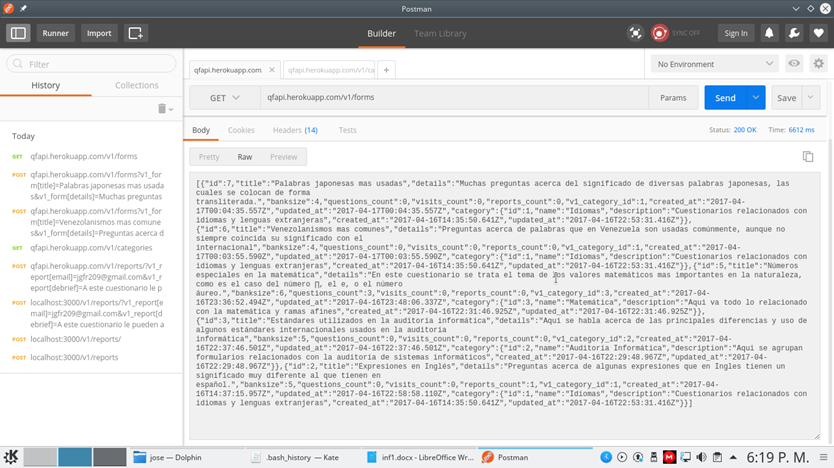


Esta vista en particular implementa el método index del modelo forms.

Si se utiliza el navegador para hacer las llamadas a la API directamente:



Y usando la aplicación Postman para visualizar los datos:



## 6.3.- Cálculo de tiempos

El tiempo invertido para cada una de las actividades desarrolladas fue el siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del backlog** | **Tareas** | **Estimación inicial (H)** | **Tiempo utilizado (H)** | **Diferencia** |
| Preparar la API para su servicio en red | Crear el repositorio de GitHub con el código del proyecto | 5 | 5 | 0 |
| Configurar y cargar el repositorio  local | 5 | 5 | 0 |
| Crear la cuenta  en Heroku | 5 | 5 | 0 |
| Configurar el servicio para  alojar la API | 10 | 8 | -2 |
| Cargar el código de la API a  Heroku | 25 | 30 | +5 |
| Probar con datos la API remota | 15 | 15 | 0 |
| Documentar  el desarrollo | 15 | 10 | -5 |
|  | Total | 80 | 78 | -2 |
|  |  |  |  |  |

La diferencia se justifica por las siguientes razones:

1. Configurar el servicio no es complejo, solo es necesario (en resumen, para detalles refiérase al incremento de este sprint) configurar un repositorio de GItHub, luego crear la “máquina virtual” en Heroku (la empresa las llama “dynos”, que en otros mecanismos como Docker serían los contenedores), y luego cargar el proyecto desde la rama maestra al servidor remoto.
2. La razón por la que cargar la API al servidor remoto se tomó tanto tiempo, es porque no se cargó 1 vez, sino unas 13 veces, ya que en cada carga, se producía un error (como que, por ejemplo, el modelo reports no funcionara luego de dejar la API sin uso un periodo de tiempo), y era necesario volver a cargar el proyecto.
3. No se cargaron todos los datos de prueba porque no se consideró necesario.

**6.4.- Gráfico de Burn-up**

## 6.5.- Gráfico de Burn-down

# 7.- SCRUM Retrospective

## 7.1.- Cálculo de tiempos

El tiempo invertido en todo el proyecto es el siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elementos del backlog** | **Tareas** | **Tiempo estimado inicial (H)** | **Tiempo utilizado (H)** | **Diferencia** |
| Documentar las características comerciales y técnicas de la API a ofrecer | Desarrollar el modelo CANVAS del servicio que se va a ofrecer | 5 | 3 | -2 |
| Diseñar el modelo entidad-relación del servicio a implementar | 5 | 4 | -1 |
| Desarrollar el diccionario de datos para cada una de las entidades creadas | 5 | 4 | -1 |
| Desarrollar la lista de los puntos finales del servicio a implementar | 15 | 15 | 0 |
| Documentar el uso de la aplicación Postman | 20 | 18 | -2 |
| Total | 50 | 44 | -6 |
| **Elemento del backlog** | **Tareas** | **Estimación inicial (H)** | **Tiempo utilizado (H)** | **Diferencia** |
| Implementar el comportamiento básico de la API a ofrecer | Crear los modelos base de la API que se va a ofrecer a los usuarios | 20 | 18 | -2 |
| Implementar las rutas creadas en la lista de puntos de acceso | 15 | 13 | -2 |
| Implementar las relaciones entre los modelos creados de la API | 30 | 33 | +6 |
| Documentar la creación de rutas personalizadas con Ruby on Rails | 18 | 20 | +5 |
| Crear los controladores y rutas que ofrecerán las funciones personalizadas | 17 | 17 | 0 |
| Rellenar con datos de prueba los diferentes modelos creados | 20 | 20 | 0 |
| Total | 120 | 121 | +1 |
| **Elemento del backlog** | **Tareas** | **Estimación inicial (H)** | **Tiempo utilizado (H)** | **Diferencia** |
| Preparar la API para su servicio en red | Crear el repositorio de GitHub con el código del proyecto | 5 | 5 | 0 |
| Configurar y cargar el repositorio  local | 5 | 5 | 0 |
| Crear la cuenta  en Heroku | 5 | 5 | 0 |
| Configurar el servicio para  alojar la API | 10 | 8 | -2 |
| Cargar el código de la API a  Heroku | 25 | 30 | +5 |
| Probar con datos la API remota | 15 | 15 | 0 |
| Documentar  el desarrollo | 15 | 10 | -5 |
|  | Total | 80 | 78 | -2 |
|  | **Total de horas aplicadas a la API a ofrecer** | **250** | **243** | **-7** |

## 7.2.- Gráfico de Burn-up

## 7.3.- Gráfico de Burn-down