



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Máster Universitario en Ingeniería en Informática



TFM del Máster Universitario en  
Ingeniería Informática

Análisis Visual de Revisiones de  
Código  
Documentación Técnica



Presentado por Mario Juez Gil  
en Universidad de Burgos — 30 de junio de 2017  
Tutores: Dr. Carlos López Nozal, Dr. Raúl  
Marticorena Sánchez

# Índice general

<b>Índice general</b>	<b>I</b>
<b>Índice de figuras</b>	<b>II</b>
<b>Índice de tablas</b>	<b>IV</b>
<b>Anexo A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	13
<b>Anexo B Especificación de Requisitos</b>	<b>17</b>
B.1. Introducción . . . . .	17
B.2. Catálogo de requisitos . . . . .	17
B.3. Especificación de requisitos . . . . .	21
<b>Anexo C Especificación de diseño</b>	<b>48</b>
C.1. Introducción . . . . .	48
C.2. Diseño de datos . . . . .	48
C.3. Diseño arquitectónico . . . . .	49
<b>Anexo D Documentación técnica de programación</b>	<b>63</b>
D.1. Introducción . . . . .	63
D.2. Estructura de directorios . . . . .	63
D.3. Manual del programador . . . . .	65
D.4. Compilación, instalación y ejecución del proyecto . . . . .	71
D.5. Pruebas del sistema . . . . .	74
<b>Anexo E Documentación de usuario</b>	<b>75</b>
E.1. Introducción . . . . .	75
E.2. Requisitos . . . . .	75
E.3. Manual del usuario . . . . .	76
<b>Bibliografía</b>	<b>77</b>

# Índice de figuras

A.1. Diagrama <i>burndown</i> del <i>sprint</i> 1 . . . . .	2
A.2. Diagrama <i>burndown</i> del <i>sprint</i> 2 . . . . .	3
A.3. Diagrama <i>burndown</i> del <i>sprint</i> 3 . . . . .	4
A.4. Diagrama <i>burndown</i> del <i>sprint</i> 4 . . . . .	5
A.5. Diagrama <i>burndown</i> del <i>sprint</i> 5 . . . . .	6
A.6. Diagrama <i>burndown</i> del <i>sprint</i> 6 . . . . .	7
A.7. Diagrama <i>burndown</i> del <i>sprint</i> 7 . . . . .	8
A.8. Diagrama <i>burndown</i> del <i>sprint</i> 8 . . . . .	9
A.9. Diagrama <i>burndown</i> del <i>sprint</i> 9 . . . . .	10
A.10. Diagrama <i>burndown</i> del <i>sprint</i> 10 . . . . .	10
A.11. Diagrama <i>burndown</i> del <i>sprint</i> 11 . . . . .	11
A.12. Diagrama <i>burndown</i> del <i>sprint</i> 12 . . . . .	12
A.13. Diagrama <i>burndown</i> del <i>sprint</i> 13 . . . . .	13
B.1. Diagrama de casos de uso de la aplicación. . . . .	21
C.1. Diagrama entidad-relación de datos. . . . .	49
C.2. División en capas de la aplicación. . . . .	49
C.3. Diagrama de paquetes de la aplicación. . . . .	51
C.4. Diagrama de clases de repositorios. . . . .	52
C.5. Diagrama de clases de entidades. . . . .	53
C.6. Diagrama de clases de tareas (1). . . . .	54
C.7. Diagrama de clases de tareas (2). . . . .	55
C.8. Diagrama de clases de servicios. . . . .	56
C.9. Ejemplo de diagrama de secuencia cliente-API. . . . .	57
C.10. Ejemplo de diagrama de secuencia de ejecución de una tarea. . . . .	58
C.11. Pantalla de inicio. . . . .	59
C.12. Pantalla de listado de repositorios. . . . .	59
C.13. Pantalla de análisis visual de repositorio. . . . .	60
C.14. Pantalla de listado de <i>pull requests</i> . . . . .	60
C.15. Pantalla de análisis visual de <i>pull request</i> . . . . .	61
C.16. Pantalla de listado de usuarios. . . . .	61
C.17. Pantalla de análisis visual de usuario. . . . .	62

D.1. GitHub Developer Application. . . . .	65
D.2. Visual Studio Code. . . . .	66
D.3. Creando una aplicación en Heroku. . . . .	66
D.4. Seleccionando un <i>buildpack</i> en Heroku. . . . .	67
D.5. Conectando el repositorio con Travis. . . . .	68
D.6. Configuración de <code>.travis.yml</code> . . . . .	68
D.7. Ejemplo de estado de construcción en Travis. . . . .	69
D.8. Selección de repositorio para realizar revisiones con Codebeat. . . . .	70
D.9. Codebeat como <i>check</i> de GitHub. . . . .	70
D.10. Variables de entorno en Heroku. . . . .	72
D.11. Compilación de la aplicación. . . . .	73
D.12. Ejecución de la aplicación. . . . .	73
D.13. Pruebas unitarias del sistema. . . . .	74
E.1. Fragmento de la <i>Wiki</i> en GitHub. . . . .	76

# Índice de tablas

A.1. Equivalencia <i>story-point</i> a tiempo . . . . .	2
A.2. Costes de personal a media jornada. . . . .	14
A.3. Costes de personal a jornada completa. . . . .	14
A.4. Otros costes. . . . .	15
A.5. Costes totales. . . . .	15
A.6. Diferentes licencias utilizadas por nuestras dependencias. . . . .	16
B.1. CU-01 Solicitar datos de repositorio . . . . .	22
B.2. CU-02 Listar repositorios . . . . .	23
B.3. CU-03 Análisis visual repositorio . . . . .	24
B.4. CU-04 Descarga CSV con datos de revisiones . . . . .	25
B.5. CU-05 Listar <i>pull requests</i> . . . . .	26
B.6. CU-06 Análisis visual <i>pull request</i> . . . . .	27
B.7. CU-07 Listar usuarios . . . . .	28
B.8. CU-08 Análisis visual usuario . . . . .	29
B.9. CU-09 Ver estado API . . . . .	30
B.10.CU-10 Crear tarea de obtención de datos . . . . .	31
B.11.CU-11 Obtener datos de GitHub . . . . .	32
B.12.CU-12 Obtener estado del gestor de tareas . . . . .	33
B.13.CU-13 Obtener listado de tareas . . . . .	34
B.14.CU-14 Obtener listado de <i>pull requests</i> . . . . .	35
B.15.CU-15 Obtener <i>pull request</i> . . . . .	36
B.16.CU-16 Obtener estadísticas de <i>pull request</i> . . . . .	37
B.17.CU-17 Obtener listado de usuarios . . . . .	38
B.18.CU-18 Obtener usuario . . . . .	39
B.19.CU-19 Obtener estadísticas de usuarios . . . . .	40
B.20.CU-20 Obtener estadísticas de revisiones . . . . .	41
B.21.CU-21 Obtener estadísticas de comentarios de revisión . . . . .	42
B.22.CU-22 Obtener listado de repositorios . . . . .	43
B.23.CU-23 Obtener repositorio . . . . .	44
B.24.CU-24 Obtener estadísticas de repositorios . . . . .	45
B.25.CU-25 Obtener listado de nombres de repositorios . . . . .	46
B.26.CU-26 Obtener datos CSV de repositorio . . . . .	47

## Anexo A

# Plan de Proyecto Software

### A.1. Introducción

En este anexo se tratan dos puntos principales:

- Planificación temporal del proyecto, donde se han utilizado metodologías ágiles.
- Estudio de la viabilidad del proyecto tanto a nivel económico como legal.

### A.2. Planificación temporal

La planificación temporal del proyecto se caracteriza por el uso de metodologías ágiles, concretamente Scrum. El desarrollo se ha dividido en una serie de iteraciones (o *sprints*).

Se han realizado un total de 14 *sprints*. Cada uno de ellos está enfocado a una serie de objetivos particulares. Cada *sprint* se corresponde con una *milestone* del repositorio en GitHub. A su vez, cada *milestone* está compuesta por una serie de tareas (o *issues*).

Cada una de las iteraciones finaliza con una reunión presencial o mediante videoconferencia donde se revisan y aceptan los cambios. Tras ello, los cambios aceptados son integrados en la rama principal (rama *master*).

El *add-on* ZenHub se ha utilizado para asignar estimaciones temporales a las tareas, las cuales permiten el desarrollo de un diagrama *burndown* por *sprint*. Para las estimaciones de tiempo se utiliza la unidad *story point*, a continuación se muestra una tabla de equivalencias aproximadas:

<i>Story points</i>	Tiempo
1	30 minutos
2	1 hora
3	2 horas
4	3 horas
5	6 horas
6	8 horas
8	10 horas
13	16 horas

Tabla A.1: Equivalencia *story-point* a tiempo.

A continuación se muestran las iteraciones del proyecto:

### **Sprint 1 (10/02/2017 - 24/02/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el aprendizaje de conceptos y herramientas. Sus dos tareas suman una estimación de 20 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/1](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/1)

Figura A.1: Diagrama burndown del *sprint 1*.

### **Sprint 2 (24/02/2017 - 10/03/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el desarrollo de la memoria y prototipos previos. Sus 4 tareas suman una estimación de 22 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/2](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/2)

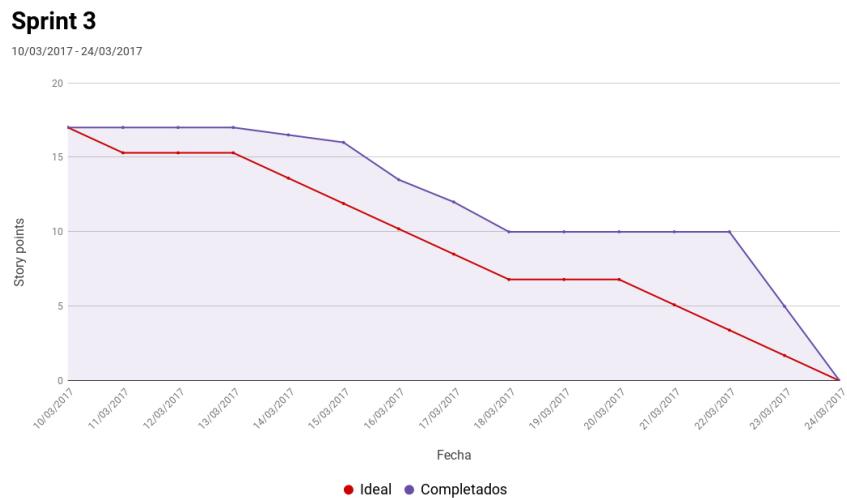


Figura A.2: Diagrama burndown del *sprint 2*.

### **Sprint 3 (10/03/2017 - 24/03/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el desarrollo de la memoria, prototipos previos y estudio de tecnologías a utilizar. Sus 5 tareas suman una estimación de 17 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/3](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/3)

Figura A.3: Diagrama burndown del *sprint 3*.

#### **Sprint 4 (24/03/2017 - 07/04/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el desarrollo de la memoria, prototipos previos y estudio de herramientas de revisión automática. Sus 6 tareas suman una estimación de 18 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/4](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/4)

Figura A.4: Diagrama burndown del *sprint 4*.

### **Sprint 5 (07/04/2017 - 21/04/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el estudio de la API de GitHub y el desarrollo de la parte *backend* de la aplicación final. Sus 8 tareas suman una estimación de 32 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/5](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/5)

Figura A.5: Diagrama burndown del *sprint 5*.

### **Sprint 6 (21/04/2017 - 28/04/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el estudio de la de las revisiones en GitHub, desarrollo de documentación, desarrollo de la parte *backend* de la aplicación final, y creación de pruebas unitarias. Sus 7 tareas suman una estimación de 24 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/6](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/6)

Figura A.6: Diagrama burndown del *sprint 6*.

### **Sprint 7 (28/04/2017 - 12/05/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el desarrollo y finalización de la parte *backend* de la aplicación final. Sus 9 tareas suman una estimación de 34 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/7](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/7)



Figura A.7: Diagrama burndown del sprint 7.

### Sprint 8 (12/05/2017 - 19/05/2017)

Esta iteración del proyecto contiene tareas relacionadas con refactorizaciones de la parte *backend* e inicio de la parte *frontend*. Sus 4 tareas suman una estimación de 20 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/8](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/8)



Figura A.8: Diagrama burndown del *sprint 8*.

### **Sprint 9 (19/05/2017 - 26/05/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el desarrollo de la parte *frontend* y alguna mejora en la parte *backend*. Sus 5 tareas suman una estimación de 20 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/9](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/9)

Figura A.9: Diagrama burndown del *sprint 9*.**Sprint 10 (26/05/2017 - 02/06/2017)**

Esta iteración del proyecto contiene tareas relacionadas con el desarrollo de la parte *frontend*. Sus 6 tareas suman una estimación de 33 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/10](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/10)

Figura A.10: Diagrama burndown del *sprint 10*.

### **Sprint 11 (02/06/2017 - 09/06/2017)**

Esta iteración del proyecto contiene tareas relacionadas con la finalización del desarrollo de la parte *frontend* y alguna mejora en la parte *backend*. Sus 8 tareas suman una estimación de 23 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/11](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/11)



Figura A.11: Diagrama burndown del *sprint 11*.

### **Sprint 12 (09/06/2017 - 19/06/2017)**

Esta iteración del proyecto contiene tareas relacionadas con la implementación de las últimas características de la aplicación final, documentación de código, y desarrollo de pruebas unitarias. Sus 7 tareas suman una estimación de 34 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/12](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/12)



Figura A.12: Diagrama burndown del *sprint 12*.

### Sprint 13 (19/06/2017 - 26/06/2017)

Esta iteración del proyecto contiene tareas relacionadas con la finalización del desarrollo de la memoria del proyecto. Sus 11 tareas suman una estimación de 44 *story points*.

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/milestone/13](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/13)

Figura A.13: Diagrama burndown del *sprint 13*.

### Sprint 14 (26/06/2017 - 03/07/2017)

Ésta es la última iteración del proyecto. Sus tareas están relacionadas con el desarrollo de los anexos (documentación técnica).

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Código/milestone/14](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/milestone/14)

En el momento de realización de este documento, la iteración aún no había terminado, y por tanto no existía el diagrama *burndown*.

## A.3. Estudio de viabilidad

A continuación se desarrolla el estudio de viabilidad del proyecto, tanto a nivel económico como a nivel legal.

### Análisis coste-beneficio

El estudio de viabilidad económica consiste en el análisis de costes y beneficios del proyecto.

#### Costes

Los costes del proyecto se han desglosado en los siguientes tipos:

#### Costes de personal:

El proyecto ha sido realizado por un desarrollador empleado a media jornada durante 3 meses, y a jornada completa durante 2 meses:

Descripción	Coste
Salario mensual neto	550,00 €
Retención IRPF (19 %)	204,50 €
Seguridad Social (29,9 %)	321,82 €
Salario mensual bruto	1.076,32 €
<b>Total 3 meses</b>	<b>3.228,96 €</b>

Tabla A.2: Costes de personal a media jornada.

Descripción	Coste
Salario mensual neto	1.100,00 €
Retención IRPF (19 %)	409,00 €
Seguridad Social (29,9 %)	643,64 €
Salario mensual bruto	2.152,64 €
<b>Total 2 meses</b>	<b>4.305,28 €</b>

Tabla A.3: Costes de personal a jornada completa.

Los costes de Seguridad Social (29,9 %) se han calculado tal como indica el Ministerio de Empleo y Seguridad Social [2]:

- 23,60 % por contingencias comunes.
- 5,50 % por desempleo de tipo general.
- 0,20 % para el Fondo de Garantía Salarial.
- 0,60 % por formación profesional.

#### Costes de hardware:

Para el desarrollo del proyecto se ha utilizado un ordenador portátil con un coste de 990 € y una vida útil de 5 años. El coste amortizado es de 82,5 €.

Para calcular el coste de amortización se ha utilizado la siguiente fórmula:

$$\text{CosteAmortizado} = \frac{\text{CosteHardware}}{\text{VidaUtil}} * \text{TiempoUtilizado}$$

**Costes de software:**

Todo el *software* utilizado durante el desarrollo del proyecto es gratuito o tiene planes gratuitos.

**Otros costes:**

A continuación se detalla el resto de costes del proyecto:

Descripción	Coste
Internet	165,25 €
Documentación (memoria impresa + CD's)	50,00 €
<b>Total</b>	<b>215,25 €</b>

Tabla A.4: Otros costes.

**Costes totales:**

La suma de todos los costes del proyecto es la siguiente:

Descripción	Coste
Personal	7.534,24 €
<i>Hardware</i>	82,50 €
<i>Software</i>	0,00 €
Otros	215,25 €
<b>Total</b>	<b>7.831,99 €</b>

Tabla A.5: Costes totales.

**Beneficios**

Este proyecto se ha desarrollado para ser utilizado en entornos de investigación universitaria. No se espera obtener beneficio económico del mismo.

**Viabilidad legal**

El estudio de la viabilidad legal se ha realizado en términos relativos a las licencias utilizadas.

### Licencia del *software* desarrollado

Como se ha indicado anteriormente, nuestro proyecto tiene fines académicos y de investigación. Por ello no queremos restringir su uso, distribución o modificación.

Para la elección de la licencia de la aplicación desarrollada se han tenido en cuenta las licencias de las dependencias externas utilizadas.

Licencia	Tipo
WTFPL	Copyright
MIT	Copyright
ISC	Copyright
Apache 2.0	Copyright
BSD-3-Clause	Copyright

Tabla A.6: Diferentes licencias utilizadas por nuestras dependencias.

La licencia MIT [4] se ajusta a nuestras necesidades. Es compatible con todas las anteriores, y además es poco restrictiva. Permite, entre otras cosas, el uso, copia, modificación o distribución del *software* con la única condición de incluir nuestro aviso de copyright y licencia en toda copia o derivado.

La licencia GPLv3 también era compatible con las licencias de nuestras dependencias, pero al ser de tipo copyleft, es algo más restrictiva que la MIT.

### Licencia de la documentación

En lo relativo a la documentación (memoria y anexos), se ha utilizado una licencia de tipo Creative Commons 4.0 [1] (CC BY 4.0), que permite la copia y redistribución, así como su modificación para cualquier propósito.

## Anexo B

# Especificación de Requisitos

### B.1. Introducción

Este anexo está dedicado a la especificación de requisitos del proyecto. Se distingue entre requisitos funcionales y requisitos no funcionales y se utilizarán diagramas de caso de uso.

La aplicación está dividida en dos subsistemas, un cliente web y una API REST, para cada uno de los cuales se ha desarrollado una especificación de requisitos particular.

### B.2. Catálogo de requisitos

En esta parte se definen los diferentes requisitos de ambos subsistemas del proyecto.

#### Requisitos funcionales del cliente web

- **RFCW-1 Solicitud de datos de un repositorio:** la aplicación debe permitir la solicitud de los datos de revisiones de un repositorio de GitHub.
- **RFCW-2 Listado de repositorios:** se deben listar todos los repositorios disponibles en la aplicación. El listado debe ofrecer posibilidades de ordenación.
- **RFCW-3 Análisis visual de datos de un repositorio:** debe existir una vista que contenga gráficos creados a partir de datos sobre las revisiones de código realizadas en un repositorio.
- **RFCW-4 Descarga fichero CSV de un repositorio:** el usuario debe ser capaz de descargar un fichero .CSV que contenga datos sobre las *pull requests* y revisiones realizadas en un repositorio concreto.

- **RFCW-5 Listado de *pull requests*:** se deben listar todas las *pull requests* disponibles en la aplicación. El listado debe ofrecer posibilidades de ordenación y de filtrado (por repositorio).
- **RFCW-6 Análisis visual de datos de una *pull request*:** debe existir una vista que contenga gráficos creados a partir de datos de una *pull request*.
- **RFCW-7 Listado de usuarios:** se deben listar todos los usuarios disponibles en la aplicación. El listado debe ofrecer posibilidades de ordenación.
- **RFCW-8 Análisis visual de datos de un usuario:** debe existir una vista que contenga gráficos creados a partir de datos sobre las revisiones de código realizadas por un usuario (revisor).
- **RFCW-9 Visualización de estado de API:** el cliente web debe mostrar un indicador con el estado actual de la API para conocer si se encuentra obteniendo datos o no.

### Requisitos no funcionales del cliente web

- **RNFCW-1 Usabilidad:** el cliente web debe tener una interfaz intuitiva y fácil de utilizar.
- **RNFCW-2 Rendimiento:** el cliente web debe evitar los refrescos de página y optimizar el número de peticiones HTTP necesarias.
- **RNFCW-3 Portabilidad:** el cliente web debe funcionar correctamente en los tres principales navegadores web: Edge, Firefox y Chrome.

### Requisitos funcionales de la API REST

- **RFAR-1 Obtención de entidades de GitHub:** la API REST debe ser capaz de obtener todo tipo de entidad relacionada con las revisiones de código y almacenarlas para su posterior uso.
- **RFAR-2 Obtención del estado del gestor de tareas:** debe existir una ruta de la API que permita conocer el estado actual del gestor de tareas de obtención de datos.
- **RFAR-3 Obtención del listado de tareas:** debe existir una ruta en la API para obtener el listado de tareas. Dicha ruta debe contener un parámetro que permita escoger si se desean obtener todas las tareas (incluidas las completas) o únicamente las incompletas.

- **RFAR-4 Obtención del listado de *pull requests*:** debe existir una ruta en la API para obtener el listado de entidades de tipo *pull request*. Mediante la parametrización de la ruta se debe permitir la ordenación y filtrado (por repositorio) de la lista.
- **RFAR-5 Obtención de una *pull request*:** debe existir una ruta en la API para obtener los datos de una entidad de tipo *pull request*.
- **RFAR-6 Obtención de estadísticas de *pull requests*:** debe existir una ruta en la API para obtener los valores medios de propiedades de una lista de *pull requests* filtradas por usuario o por repositorio. Parametrizando la ruta también se debe tener acceso a estadísticas temporales para un usuario o repositorio concreto. Éstas se deben calcular en un máximo de 20 puntos temporales equidistantes desde la fecha de creación de la *pull request* más antigua de la lista hasta la fecha de creación más reciente de la lista.
- **RFAR-7 Obtención del listado de usuarios:** debe existir una ruta en la API para obtener el listado de entidades de tipo usuario. Mediante la parametrización de la ruta se debe permitir la ordenación de la lista.
- **RFAR-8 Obtención de un usuario:** debe existir una ruta en la API para obtener los datos de una entidad de tipo usuario.
- **RFAR-9 Obtención de estadísticas de usuario:** debe existir una ruta en la API para obtener los valores medios de propiedades de la lista de todas las entidades usuario de la aplicación.
- **RFAR-10 Obtención de estadísticas de revisiones:** debe existir una ruta en la API para obtener estadísticas sobre una lista de revisiones de un repositorio o usuario concreto. Las estadísticas se deben calcular en un máximo de 20 puntos temporales equidistantes desde la fecha de creación de la revisión más antigua de la lista hasta la fecha de creación más reciente de la lista.
- **RFAR-11 Obtención de estadísticas de comentarios de revisión:** debe existir una ruta en la API para obtener estadísticas sobre una lista de comentarios de revisión de un repositorio o usuario concreto. Las estadísticas se deben calcular en un máximo de 20 puntos temporales equidistantes desde la fecha de creación del comentario de revisión más antiguo de la lista hasta la fecha de creación más reciente de la lista.
- **RFAR-12 Obtención del listado de repositorios:** debe existir una ruta en la API para obtener el listado de entidades de tipo repositorio. Mediante la parametrización de la ruta se debe permitir la ordenación de la lista.

- **RFAR-13 Obtención de un repositorio:** debe existir una ruta en la API para obtener los datos de una entidad de tipo repositorio.
- **RFAR-14 Obtención de estadísticas de repositorio:** debe existir una ruta en la API para obtener los valores medios de propiedades de la lista de todas las entidades repositorio de la aplicación.
- **RFAR-15 Obtención de un listado de nombres de repositorios:** debe existir una ruta en la API para obtener el listado con todos los nombres de repositorio almacenados en la aplicación.
- **RFAR-16 Obtención de un fichero CSV con datos sobre un repositorio:** debe existir una ruta en la API para obtener un fichero .CSV con datos de revisiones de código y *pull requests* de un repositorio concreto.

### Requisitos no funcionales de la API REST

- **RNFAR-1 Escalabilidad:** la arquitectura de la API REST debe ser escalable para que añadir soporte a otro tipo de entidades de GitHub en el futuro resulte fácil.
- **RNFAR-2 Rendimiento:** la API REST debe funcionar correctamente en una máquina con un mínimo de 500 MB de memoria RAM.
- **RNFAR-3 Seguridad:** se debe evitar encolar el mismo repositorio más de una vez, ya que en caso contrario se podrían dar casos de inanición. También se debe evitar encolar repositorios inexistentes.
- **RNFAR-4 Mantenibilidad:** el código de la API debe tener una calidad mínima de 3.0 GPA para asegurar su mantenibilidad.

### B.3. Especificación de requisitos

En esta parte se muestra el diagrama de casos de uso del proyecto y la definición de cada uno de ellos.

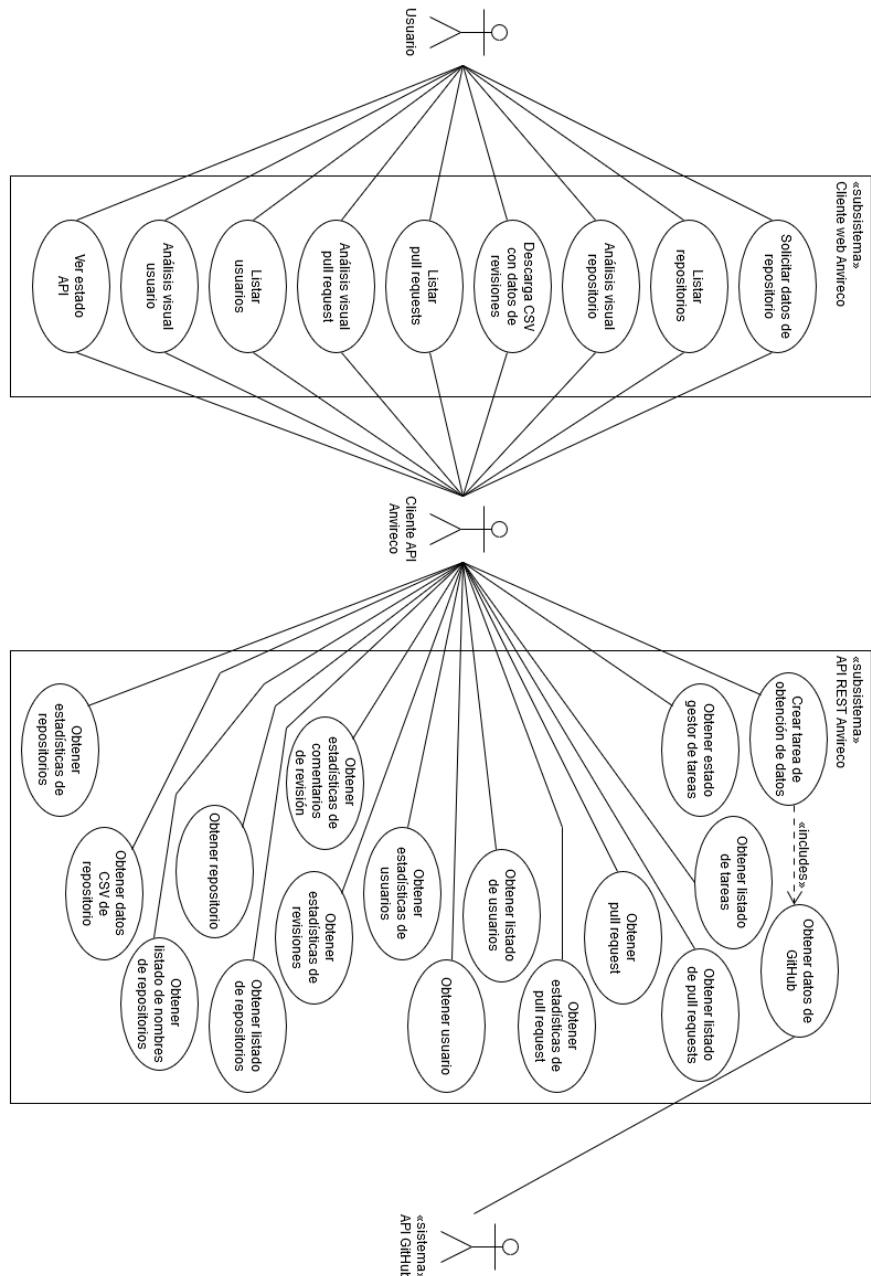


Figura B.1: Diagrama de casos de uso de la aplicación.

CU-01	<b>Solicitar datos de repositorio</b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFCW-1
<b>Descripción</b>	Permite al usuario solicitar la obtención de datos de revisiones de un repositorio de GitHub.
<b>Precondición</b>	Situado en la página de inicio del cliente web.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1      Rellenar el campo propietario.</li> <li>2      Rellenar el campo repositorio.</li> <li>3      Pulsar el botón “adelante”.</li> </ol>
<b>Postcondición</b>	El número de tareas de obtención de datos en el servidor se ha incrementado.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1      Repositorio inexistente. Se muestra un mensaje informativo.</li> <li>2      API no disponible. Se muestra un mensaje informativo.</li> </ol>
<b>Importancia</b>	Alta

Tabla B.1: CU-01 Solicitar datos de repositorio .

CU-02	Listar repositorios
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFCW-2
<b>Descripción</b>	Permite al usuario ver una lista de todos los repositorios almacenados en la aplicación.
<b>Precondición</b>	Cliente web Anvireco abierto.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	1       Pulsar el elemento “Repositorios” del menú lateral.
	2       Opcionalmente se pueden ordenar los repositorios utilizando el control desplegable “Orden...” de la vista.
<b>Postcondición</b>	El tamaño de la lista es de 100 elementos o se corresponde con el número de repositorios almacenados en la base de datos en caso contrario.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	1       API no disponible. Se muestra un mensaje informativo.
<b>Importancia</b>	Alta

Tabla B.2: CU-02 Listar repositorios .

<b>CU-03</b>		<b>Análisis visual repositorio</b>
<b>Versión</b>		1.0
<b>Autor</b>		Mario Juez Gil
<b>Requisitos asociados</b>		RFCW-3
<b>Descripción</b>		Permite al usuario ver una serie de gráficos sobre datos generales y datos de revisiones de código de un repositorio.
<b>Precondición</b>		Situado en la vista de listado de repositorios.
		<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	1	Pulsar en el nombre del repositorio objetivo.
	2	Opcionalmente se puede navegar al listado de <i>pull requests</i> de ese repositorio.
	3	Opcionalmente se puede navegar a la página del repositorio en GitHub.
<b>Postcondición</b>		Los gráficos mostrados se corresponden con los datos del repositorio elegido.
		<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	1	API no disponible. Se muestra un mensaje informativo.
<b>Importancia</b>		Alta

Tabla B.3: CU-03 Análisis visual repositorio .

CU-04	Descarga CSV con datos de revisiones
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFCW-4
<b>Descripción</b>	Permite al usuario descargar un fichero .CSV con datos sobre <i>pull requests</i> y revisiones del repositorio.
<b>Precondición</b>	Situado en la vista de un repositorio concreto.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	1       Pulsar sobre el botón de descarga de CSV.
<b>Postcondición</b>	Se muestra un diálogo de descarga del fichero .CSV. El fichero contiene los datos del repositorio escogido.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	1       API no disponible. Se muestra un mensaje informativo.
<b>Importancia</b>	Media

Tabla B.4: CU-04 Descarga CSV con datos de revisiones .

CU-05	Listar <i>pull requests</i>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFCW-5
<b>Descripción</b>	Permite al usuario ver una lista de todas las <i>pull requests</i> almacenadas en la aplicación.
<b>Precondición</b>	Cliente web Anvireco abierto.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	<p>1 Pulsar el elemento “Pull Requests” del menú lateral.</p> <p>2 Opcionalmente se pueden ordenar las <i>pull requests</i> utilizando el control desplegable “Orden...” de la vista.</p> <p>3 Opcionalmente se pueden filtrar las <i>pull requests</i> por repositorio utilizando el control desplegable “Filtrar por repositorio” de la vista.</p>
<b>Postcondición</b>	El tamaño de la lista es de 100 elementos o se corresponde con el número de <i>pull requests</i> almacenadas en la base de datos en caso contrario.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	<p>1 API no disponible. Se muestra un mensaje informativo.</p>
<b>Importancia</b>	Alta

Tabla B.5: CU-05 Listar *pull requests* .

CU-06	Análisis visual <i>pull request</i>				
<b>Versión</b>	1.0				
<b>Autor</b>	Mario Juez Gil				
<b>Requisitos asociados</b>	RFCW-6				
<b>Descripción</b>	Permite al usuario ver una serie de gráficos sobre datos generales de una <i>pull request</i> .				
<b>Precondición</b>	Situado en la vista de listado de <i>pull requests</i> .				
	<table border="1"> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> </table>	Paso	Acción		
Paso	Acción				
<b>Acciones</b>	<table border="1"> <tr> <td>1</td> <td>Pulsar en el nombre de la <i>pull request</i> objetivo.</td> </tr> <tr> <td>2</td> <td>Opcionalmente se puede navegar a la página de la <i>pull request</i> en GitHub.</td> </tr> </table>	1	Pulsar en el nombre de la <i>pull request</i> objetivo.	2	Opcionalmente se puede navegar a la página de la <i>pull request</i> en GitHub.
1	Pulsar en el nombre de la <i>pull request</i> objetivo.				
2	Opcionalmente se puede navegar a la página de la <i>pull request</i> en GitHub.				
<b>Postcondición</b>	Los gráficos mostrados se corresponden con los datos de la <i>pull request</i> elegida.				
	<table border="1"> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> </table>	Paso	Acción		
Paso	Acción				
<b>Excepciones</b>	<table border="1"> <tr> <td>1</td> <td>API no disponible. Se muestra un mensaje informativo.</td> </tr> </table>	1	API no disponible. Se muestra un mensaje informativo.		
1	API no disponible. Se muestra un mensaje informativo.				
<b>Importancia</b>	Alta				

Tabla B.6: CU-06 Análisis visual *pull request* .

CU-07	Listar usuarios
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFCW-7
<b>Descripción</b>	Permite al usuario ver una lista de todos los usuarios almacenadas en la aplicación.
<b>Precondición</b>	Cliente web Anvireco abierto.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	<p>1      Pulsar el elemento “Usuarios” del menú lateral.</p> <p>2      Opcionalmente se pueden ordenar los usuarios utilizando el control desplegable “Orden...” de la vista.</p>
<b>Postcondición</b>	El tamaño de la lista es de 100 elementos o se corresponde con el número de usuarios almacenados en la base de datos en caso contrario.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	1      API no disponible. Se muestra un mensaje informativo.
<b>Importancia</b>	Alta

Tabla B.7: CU-07 Listar usuarios .

<b>CU-08</b>		<b>Análisis visual usuario</b>
<b>Versión</b>		1.0
<b>Autor</b>		Mario Juez Gil
<b>Requisitos asociados</b>		RFCW-8
<b>Descripción</b>		Permite al usuario ver una serie de gráficos sobre datos generales y datos de revisiones de un usuario.
<b>Precondición</b>		Situado en la vista de listado de usuarios.
		<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	1	Pulsar en el nombre del usuario objetivo.
	2	Opcionalmente se puede navegar a la página del usuario en GitHub.
<b>Postcondición</b>		Los gráficos mostrados se corresponden con los datos del usuario elegido.
		<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	1	API no disponible. Se muestra un mensaje informativo.
<b>Importancia</b>		Alta

Tabla B.8: CU-08 Análisis visual usuario .

CU-09	Ver estado API
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFCW-9
<b>Descripción</b>	Permite al usuario ver el estado actual de la API.
<b>Precondición</b>	Cliente web Anvireco abierto.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	<p>1      El estado se muestra en la parte inferior de la aplicación.</p>
<b>Postcondición</b>	Estado visible se corresponde con el estado del gestor de tareas.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	<p>1      API no disponible. No se muestra el estado.</p>
<b>Importancia</b>	Baja

Tabla B.9: CU-09 Ver estado API .

CU-10	Crear tarea de obtención de datos
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFAR-1
<b>Descripción</b>	Permite al usuario encolar una tarea para obtener datos de revisiones de un repositorio de GitHub.
<b>Precondición</b>	API REST en ejecución.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	<p>1      Petición POST a la ruta de creación de tareas parametrizada con el propietario y nombre del repositorio.</p>
<b>Postcondición</b>	Existen 8 tareas más en la base de datos.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	<p>1      El repositorio no existe en GitHub. No se crea ninguna tarea.</p> <p>2      La API de GitHub no está disponible. No se crea ninguna tarea.</p> <p>3      La API de Anvireco no está disponible. No se crea ninguna tarea.</p>
<b>Importancia</b>	Alta

Tabla B.10: CU-10 Crear tarea de obtención de datos .

CU-11	Obtener datos de GitHub
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFAR-1
<b>Descripción</b>	Obtención de datos de entidades relacionadas con revisiones de código de GitHub.
<b>Precondición</b>	API REST en ejecución. API GitHub disponible.
Paso	Acción
<b>Acciones</b>	<p>1 Ejecución de la cola de tareas de obtención de datos pendientes.</p>
<b>Postcondición</b>	El número de entidades almacenadas en nuestra base de datos se ha incrementado.
Paso	Acción
<b>Excepciones</b>	<p>1 La API de GitHub no está disponible. El gestor de tareas se pausa para reintentar la obtención más adelante.</p> <p>2 La base de datos de Anvireco no está disponible. El gestor de tareas se pausa para reintentar la obtención más adelante.</p> <p>3 Se ha excedido el límite de peticiones a la API de GitHub. El gestor de tareas se pausa para reintentar la obtención más adelante.</p>
<b>Importancia</b>	Alta

Tabla B.11: CU-11 Obtener datos de GitHub .

<b>CU-12</b>	<b>Obtener estado del gestor de tareas</b>				
<b>Versión</b>	1.0				
<b>Autor</b>	Mario Juez Gil				
<b>Requisitos asociados</b>	RFAR-2				
<b>Descripción</b>	Permite al usuario conocer el estado actual del gestor de tareas de obtención de datos.				
<b>Precondición</b>	API REST en ejecución.				
<b>Acciones</b>	<table border="1"> <thead> <tr> <th><b>Paso</b></th><th><b>Acción</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>Petición GET a la ruta del estado del gestor de tareas.</td></tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	Petición GET a la ruta del estado del gestor de tareas.
<b>Paso</b>	<b>Acción</b>				
1	Petición GET a la ruta del estado del gestor de tareas.				
<b>Postcondición</b>	El estado mostrado en la respuesta se corresponde con el estado del gestor de tareas en ese momento.				
<b>Excepciones</b>	<table border="1"> <thead> <tr> <th><b>Paso</b></th><th><b>Acción</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>La API de Anvireco no está disponible. No hay respuesta.</td></tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	La API de Anvireco no está disponible. No hay respuesta.
<b>Paso</b>	<b>Acción</b>				
1	La API de Anvireco no está disponible. No hay respuesta.				
<b>Importancia</b>	Baja				

Tabla B.12: CU-12 Obtener estado del gestor de tareas .

<b>CU-13</b>		<b>Obtener listado de tareas</b>
<b>Versión</b>	1.0	
<b>Autor</b>	Mario Juez Gil	
<b>Requisitos asociados</b>	RFAR-3	
<b>Descripción</b>	Permite al usuario obtener un listado con las tareas de la aplicación en formato JSON.	
<b>Precondición</b>	API REST en ejecución.	
		<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	1	Petición GET a la ruta del listado de tareas indicando por medio de un parámetro si se quieren obtener todas las tareas, o únicamente las incompletas.
<b>Postcondición</b>		El tamaño del listado de tareas se corresponde con el número de tareas almacenadas (y filtradas si fuese el caso) en la base de datos.
		<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	1	La API de Anvireco no está disponible. No hay respuesta.
<b>Importancia</b>	Baja	

Tabla B.13: CU-13 Obtener listado de tareas .

CU-14	<b>Obtener listado de <i>pull requests</i></b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFAR-4
<b>Descripción</b>	Permite al usuario obtener un listado con las <i>pull requests</i> de la aplicación en formato JSON.
<b>Precondición</b>	API REST en ejecución.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	<p>1      Petición GET a la ruta del listado de <i>pull requests</i> indicando por medio de parámetros si el listado debe estar filtrado por repositorio y/o si debe estar ordenado de algún modo.</p>
<b>Postcondición</b>	El tamaño del listado de <i>pull requests</i> se corresponde con el número de <i>pull requests</i> almacenadas que cumplen las restricciones de filtrado y ordenación.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	<p>1      La API de Anvireco no está disponible.           No hay respuesta.</p>
<b>Importancia</b>	Alta

Tabla B.14: CU-14 Obtener listado de *pull requests* .

CU-15	<b>Obtener <i>pull request</i></b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFAR-5
<b>Descripción</b>	Permite al usuario obtener una <i>pull request</i> de la aplicación en formato JSON.
<b>Precondición</b>	API REST en ejecución.
Paso	Acción
1	Petición GET a la ruta de una <i>pull request</i> .
<b>Postcondición</b>	Los datos obtenidos de la <i>pull request</i> escogida se corresponden con los datos de la <i>pull request</i> en GitHub.
Paso	Acción
1	La API de Anvireco no está disponible. No hay respuesta.
<b>Importancia</b>	Alta

Tabla B.15: CU-15 Obtener *pull request* .

<b>CU-16</b>		<b>Obtener estadísticas de <i>pull request</i></b>
<b>Versión</b>		1.0
<b>Autor</b>		Mario Juez Gil
<b>Requisitos asociados</b>		RFAR-6
<b>Descripción</b>		Permite al usuario obtener estadísticas sobre las <i>pull requests</i> de la aplicación en formato JSON.
<b>Precondición</b>		API REST en ejecución.
		<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	1	Petición GET a la ruta de estadísticas de <i>pull requests</i> , indicando por medio de parámetros el usuario o repositorio concreto para el cual obtener las estadísticas, y si se esperan estadísticas de valores medios, o estadísticas temporales.
<b>Postcondición</b>		Los datos estadísticos obtenidos cumplen las restricciones elegidas mediante parámetros.
		<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	1	La API de Anvireco no está disponible. No hay respuesta.
<b>Importancia</b>		Alta

Tabla B.16: CU-16 Obtener estadísticas de *pull request* .

<b>CU-17</b>	<b>Obtener listado de usuarios</b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFAR-7
<b>Descripción</b>	Permite al usuario obtener un listado con los usuarios de la aplicación en formato JSON.
<b>Precondición</b>	API REST en ejecución.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	<p>1      Petición GET a la ruta del listado de usuarios indicando por medio de parámetros si el listado debe estar ordenado de algún modo.</p>
<b>Postcondición</b>	El tamaño del listado de usuarios obtenido se corresponde con el número de usuarios almacenados que cumplen las restricciones de ordenación.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	<p>1      La API de Anvireco no está disponible.           No hay respuesta.</p>
<b>Importancia</b>	Alta

Tabla B.17: CU-17 Obtener listado de usuarios .

<b>CU-18</b>	<b>Obtener usuario</b>				
<b>Versión</b>	1.0				
<b>Autor</b>	Mario Juez Gil				
<b>Requisitos asociados</b>	RFAR-8				
<b>Descripción</b>	Permite al usuario obtener un usuario de la aplicación en formato JSON.				
<b>Precondición</b>	API REST en ejecución.				
<b>Acciones</b>	<table> <thead> <tr> <th><b>Paso</b></th><th><b>Acción</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>Petición GET a la ruta de un usuario.</td></tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	Petición GET a la ruta de un usuario.
<b>Paso</b>	<b>Acción</b>				
1	Petición GET a la ruta de un usuario.				
<b>Postcondición</b>	Los datos obtenidos del usuario escogido se corresponden con los datos del usuario en GitHub.				
<b>Excepciones</b>	<table> <thead> <tr> <th><b>Paso</b></th><th><b>Acción</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>La API de Anvireco no está disponible. No hay respuesta.</td></tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	La API de Anvireco no está disponible. No hay respuesta.
<b>Paso</b>	<b>Acción</b>				
1	La API de Anvireco no está disponible. No hay respuesta.				
<b>Importancia</b>	Alta				

Tabla B.18: CU-18 Obtener usuario .

CU-19	Obtener estadísticas de usuarios	
<b>Versión</b>	1.0	
<b>Autor</b>	Mario Juez Gil	
<b>Requisitos asociados</b>	RFAR-9	
<b>Descripción</b>	Permite al usuario obtener estadísticas sobre los usuarios de la aplicación en formato JSON.	
<b>Precondición</b>	API REST en ejecución.	
Acciones	Paso	Acción
	1	Petición GET a la ruta de estadísticas de todos los usuarios.
<b>Postcondición</b>	Los datos estadísticos obtenidos se corresponden con los valores medios de las propiedades de todos los usuarios almacenados en la base de datos.	
Excepciones	Paso	Acción
	1	La API de Anvireco no está disponible. No hay respuesta.
<b>Importancia</b>	Alta	

Tabla B.19: CU-19 Obtener estadísticas de usuarios .

<b>CU-20</b>	<b>Obtener estadísticas de revisiones</b>
<b>Versión</b>	1.0
<b>Autor</b>	Mario Juez Gil
<b>Requisitos asociados</b>	RFAR-10
<b>Descripción</b>	Permite al usuario obtener estadísticas sobre las revisiones de la aplicación en formato JSON.
<b>Precondición</b>	API REST en ejecución.
	<b>Paso</b> <b>Acción</b>
<b>Acciones</b>	<p>1      Petición GET a la ruta de estadísticas de revisiones, indicando por medio de parámetros el usuario o repositorio concreto para el cual obtener las estadísticas.</p>
<b>Postcondición</b>	Los datos estadísticos obtenidos cumplen las restricciones elegidas mediante parámetros.
	<b>Paso</b> <b>Acción</b>
<b>Excepciones</b>	<p>1      La API de Anvireco no está disponible. No hay respuesta.</p>
<b>Importancia</b>	Alta

Tabla B.20: CU-20 Obtener estadísticas de revisiones .

<b>CU-21</b>	<b>Obtener estadísticas de comentarios de revisión</b>				
<b>Versión</b>	1.0				
<b>Autor</b>	Mario Juez Gil				
<b>Requisitos asociados</b>	RFAR-11				
<b>Descripción</b>	Permite al usuario obtener estadísticas sobre los comentarios de revisión de la aplicación en formato JSON.				
<b>Precondición</b>	API REST en ejecución.				
<b>Acciones</b>	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>Petición GET a la ruta de estadísticas de comentarios de revisión, indicando por medio de parámetros el usuario o repositorio concreto para el cual obtener las estadísticas.</td></tr> </tbody> </table>	Paso	Acción	1	Petición GET a la ruta de estadísticas de comentarios de revisión, indicando por medio de parámetros el usuario o repositorio concreto para el cual obtener las estadísticas.
Paso	Acción				
1	Petición GET a la ruta de estadísticas de comentarios de revisión, indicando por medio de parámetros el usuario o repositorio concreto para el cual obtener las estadísticas.				
<b>Postcondición</b>	Los datos estadísticos obtenidos cumplen las restricciones elegidas mediante parámetros.				
<b>Excepciones</b>	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>La API de Anvireco no está disponible. No hay respuesta.</td></tr> </tbody> </table>	Paso	Acción	1	La API de Anvireco no está disponible. No hay respuesta.
Paso	Acción				
1	La API de Anvireco no está disponible. No hay respuesta.				
<b>Importancia</b>	Alta				

Tabla B.21: CU-21 Obtener estadísticas de comentarios de revisión .

<b>CU-22</b>		<b>Obtener listado de repositorios</b>
<b>Versión</b>	1.0	
<b>Autor</b>	Mario Juez Gil	
<b>Requisitos asociados</b>	RFAR-12	
<b>Descripción</b>	Permite al usuario obtener un listado con los repositorios de la aplicación en formato JSON.	
<b>Precondición</b>	API REST en ejecución.	
	<b>Paso</b> <b>Acción</b>	
<b>Acciones</b>	1	Petición GET a la ruta del listado de repositorios indicando por medio de parámetros si el listado debe estar ordenado de algún modo.
<b>Postcondición</b>	El tamaño del listado de repositorios obtenido se corresponde con el número de repositorios almacenados que cumplen las restricciones de ordenación.	
	<b>Paso</b> <b>Acción</b>	
<b>Excepciones</b>	1	La API de Anvireco no está disponible. No hay respuesta.
<b>Importancia</b>	Alta	

Tabla B.22: CU-22 Obtener listado de repositorios .

CU-23	Obtener repositorio				
<b>Versión</b>	1.0				
<b>Autor</b>	Mario Juez Gil				
<b>Requisitos asociados</b>	RFAR-13				
<b>Descripción</b>	Permite al usuario obtener un repositorio de la aplicación en formato JSON.				
<b>Precondición</b>	API REST en ejecución.				
<b>Acciones</b>	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>Petición GET a la ruta de un repositorio.</td></tr> </tbody> </table>	Paso	Acción	1	Petición GET a la ruta de un repositorio.
Paso	Acción				
1	Petición GET a la ruta de un repositorio.				
<b>Postcondición</b>	Los datos obtenidos del repositorio escogido se corresponden con los datos del repositorio en GitHub.				
<b>Excepciones</b>	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>La API de Anvireco no está disponible. No hay respuesta.</td></tr> </tbody> </table>	Paso	Acción	1	La API de Anvireco no está disponible. No hay respuesta.
Paso	Acción				
1	La API de Anvireco no está disponible. No hay respuesta.				
<b>Importancia</b>	Alta				

Tabla B.23: CU-23 Obtener repositorio .

<b>CU-24</b>	<b>Obtener estadísticas de repositorios</b>				
<b>Versión</b>	1.0				
<b>Autor</b>	Mario Juez Gil				
<b>Requisitos asociados</b>	RFAR-14				
<b>Descripción</b>	Permite al usuario obtener estadísticas sobre los repositorios de la aplicación en formato JSON.				
<b>Precondición</b>	API REST en ejecución.				
<b>Acciones</b>	<table border="1"> <thead> <tr> <th><b>Paso</b></th><th><b>Acción</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>Petición GET a la ruta de estadísticas de todos los repositorios.</td></tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	Petición GET a la ruta de estadísticas de todos los repositorios.
<b>Paso</b>	<b>Acción</b>				
1	Petición GET a la ruta de estadísticas de todos los repositorios.				
<b>Postcondición</b>	Los datos estadísticos obtenidos se corresponden con los valores medios de las propiedades de todos los repositorios almacenados en la base de datos.				
<b>Excepciones</b>	<table border="1"> <thead> <tr> <th><b>Paso</b></th><th><b>Acción</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>La API de Anvireco no está disponible. No hay respuesta.</td></tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	La API de Anvireco no está disponible. No hay respuesta.
<b>Paso</b>	<b>Acción</b>				
1	La API de Anvireco no está disponible. No hay respuesta.				
<b>Importancia</b>	Alta				

Tabla B.24: CU-24 Obtener estadísticas de repositorios .

<b>CU-25</b>	<b>Obtener listado de nombres de repositorios</b>				
<b>Versión</b>	1.0				
<b>Autor</b>	Mario Juez Gil				
<b>Requisitos asociados</b>	RFAR-15				
<b>Descripción</b>	Permite al usuario obtener un listado con todos los nombres de los repositorios de la aplicación en formato JSON.				
<b>Precondición</b>	API REST en ejecución.				
<b>Acciones</b>	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>Petición GET a la ruta de listado de nombres de todos los repositorios.</td></tr> </tbody> </table>	Paso	Acción	1	Petición GET a la ruta de listado de nombres de todos los repositorios.
Paso	Acción				
1	Petición GET a la ruta de listado de nombres de todos los repositorios.				
<b>Postcondición</b>	El tamaño del listado obtenido se debe corresponder con el número de repositorios almacenados en la base de datos de la aplicación.				
<b>Excepciones</b>	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>La API de Anvireco no está disponible. No hay respuesta.</td></tr> </tbody> </table>	Paso	Acción	1	La API de Anvireco no está disponible. No hay respuesta.
Paso	Acción				
1	La API de Anvireco no está disponible. No hay respuesta.				
<b>Importancia</b>	Media				

Tabla B.25: CU-25 Obtener listado de nombres de repositorios .

CU-26	Obtener datos CSV de repositorio		
<b>Versión</b>	1.0		
<b>Autor</b>	Mario Juez Gil		
<b>Requisitos asociados</b>	RFAR-16		
<b>Descripción</b>	Permite al usuario obtener datos sobre revisiones de código y <i>pull requests</i> de un repositorio de la aplicación, en formato CSV.		
<b>Precondición</b>	API REST en ejecución.		
	<b>Paso</b> <b>Acción</b>		
<b>Acciones</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">1</td> <td>Petición GET a la ruta de obtención del fichero .CSV parametrizada con un propietario y un nombre del repositorio.</td> </tr> </table>	1	Petición GET a la ruta de obtención del fichero .CSV parametrizada con un propietario y un nombre del repositorio.
1	Petición GET a la ruta de obtención del fichero .CSV parametrizada con un propietario y un nombre del repositorio.		
<b>Postcondición</b>	Los datos CSV obtenidos se deben corresponder con los datos almacenados en la base de datos para ese repositorio.		
	<b>Paso</b> <b>Acción</b>		
<b>Excepciones</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">1</td> <td>La API de Anvireco no está disponible. No hay respuesta.</td> </tr> </table>	1	La API de Anvireco no está disponible. No hay respuesta.
1	La API de Anvireco no está disponible. No hay respuesta.		
<b>Importancia</b>	Media		

Tabla B.26: CU-26 Obtener datos CSV de repositorio .

## Anexo C

# Especificación de diseño

### C.1. Introducción

Este anexo describe aspectos sobre el diseño de la aplicación. Se expone el diseño de datos y el de la arquitectura utilizada.

### C.2. Diseño de datos

Para modelar los datos sobre revisiones de código realizadas en repositorios de GitHub se han utilizado las siguientes entidades:

- **Pull Request (*PullRequestEntity*):** se corresponde con la entidad *pull request* de GitHub. Es importante por que contiene revisiones de código y comentarios de revisión.
- **Repositorio (*RepositoryEntity*):** se corresponde con la entidad *repository* de GitHub. La mayor parte de las entidades de la aplicación están relacionadas con un repositorio.
- **Revisión (*ReviewEntity*):** se corresponde con la entidad *review* de GitHub. Está formada por un comentario general y un estado.
- **Comentario (*ReviewCommentEntity*):** se corresponde con la entidad *review comment* de GitHub. Contiene un comentario asociado a un cambio específico.
- **Usuario (*UserEntity*):** se corresponde con la entidad *user* de GitHub. Es importante por que un usuario puede realizar revisiones de código, y entonces adquiere el rol de revisor.
- **Tarea (*TaskEntity*):** no tiene correspondencia con ninguna entidad de GitHub. Sirve para definir tareas de obtención de datos de GitHub.

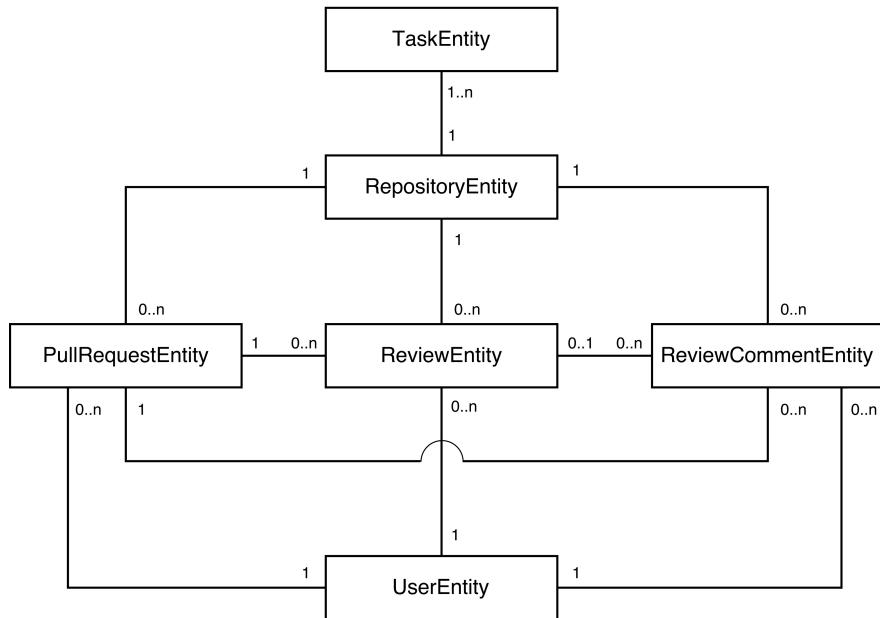


Figura C.1: Diagrama entidad-relación de datos.

### C.3. Diseño arquitectónico

A nivel arquitectónico, el servidor se ha dividido en tres capas (API REST, lógica de negocio y acceso a datos). El cliente web únicamente contiene las vistas de la aplicación.

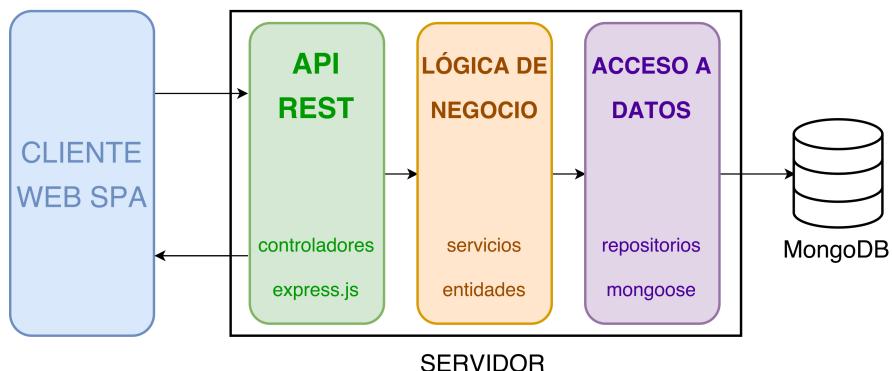


Figura C.2: División en capas de la aplicación.

## Diseño de paquetes

En JavaScript y TypeScript no existen paquetes como tal, sin embargo las clases con cometidos y responsabilidades similares se han agrupado en directorios.

La arquitectura de la aplicación sigue la siguiente estructura de paquetes.

- **src.app**: contiene las capas de acceso a datos y de lógica de negocio.
- **src.app.data**: capa de acceso a datos. Contiene una serie de clases que implementan el patrón repositorio. Incluye una carpeta **filters** con filtros para consultas y otra llamada **schemas** que contiene los *schemas* de las colecciones MongoDB.
- **src.app.entities**: pertenece a la capa de lógica de negocio. Contiene las entidades, la mayor parte de ellas son *POJOs*. Incluye una carpeta **documents** con documentos *mongoose*, y otra llamada **enum** con enumeraciones.
- **src.app.services**: pertenece a la capa de lógica de negocio. Contiene los servicios encargados de operaciones más complejas como la obtención y transformación de datos de revisiones desde GitHub. Incluye una carpeta **tasks** con implementaciones de los diferentes tipos de tareas de obtención de datos.
- **src.app.util**: contiene clases de utilidad. La mayor parte de ellas están formadas por una serie de métodos estáticos que realizan operaciones sencillas y comunes a varias clases.
- **src.client**: contiene el cliente web SPA. En la raíz se encuentran las plantillas HTML que definen el esqueleto de la aplicación.
- **src.client.css**: contiene los ficheros CSS con los estilos del cliente web.
- **src.client.js**: contiene ficheros JavaScript encargados del funcionamiento del cliente como una aplicación web *Single-Page Application*. En estos ficheros se implementa la comunicación entre el cliente y la API REST.
- **src.client.favicon**: contiene el ícono de la aplicación web para diversos dispositivos.
- **src.controllers**: capa de API REST. Contiene los controladores que gestionan las peticiones y respuestas HTTP, delegando a los servicios correspondientes.

- **src.routes**: contiene la definición de todas las rutas o *endpoints* de la API REST.
- **test.app**: contiene las pruebas unitarias de las capas de acceso a datos y lógica de negocios. Sigue la misma estructura de carpetas que **src.app**. Los ficheros de pruebas tienen el mismo nombre que la clase que se desea probar, pero con la extensión **.test.ts**.

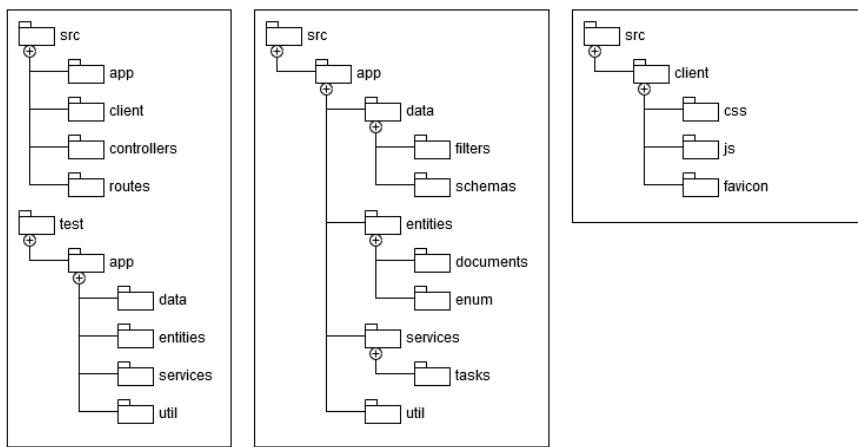


Figura C.3: Diagrama de paquetes de la aplicación.

### Diseño de clases

A continuación se muestran los diagramas de clases más relevantes de la arquitectura de la aplicación.

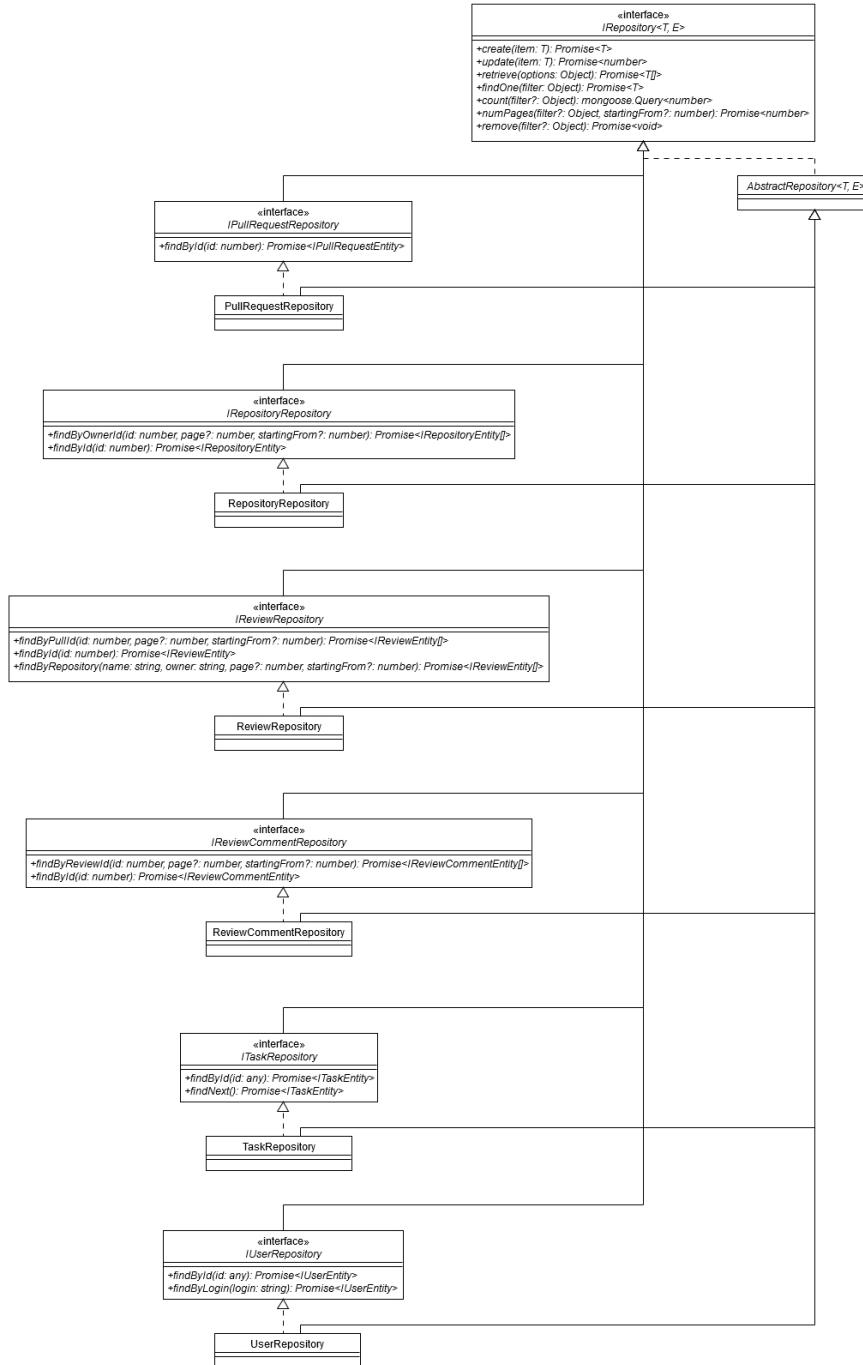


Figura C.4: Diagrama de clases de repositorios.

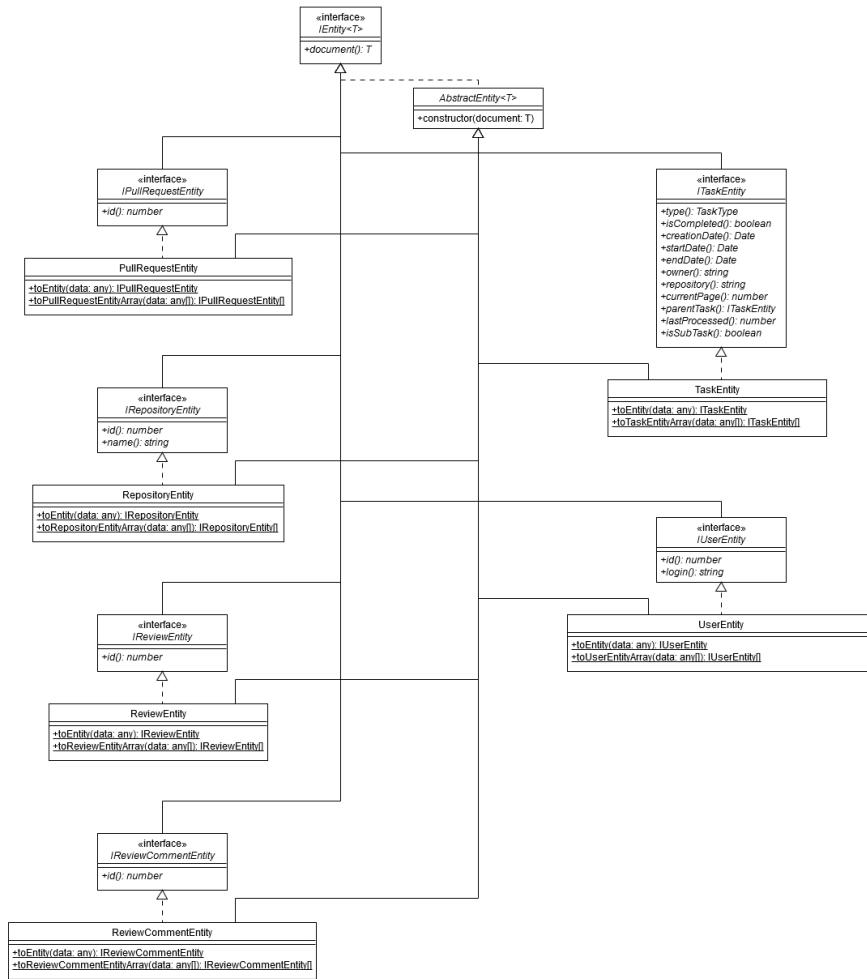


Figura C.5: Diagrama de clases de entidades.

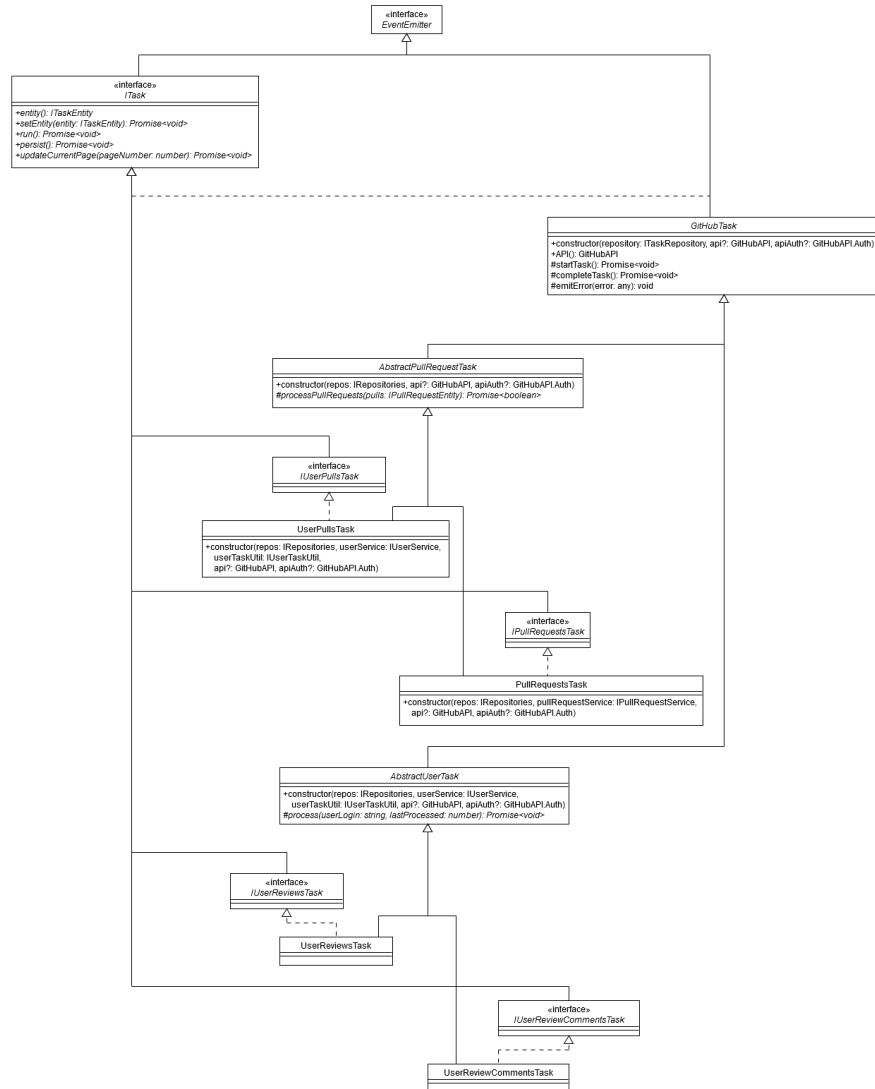


Figura C.6: Diagrama de clases de tareas (1).

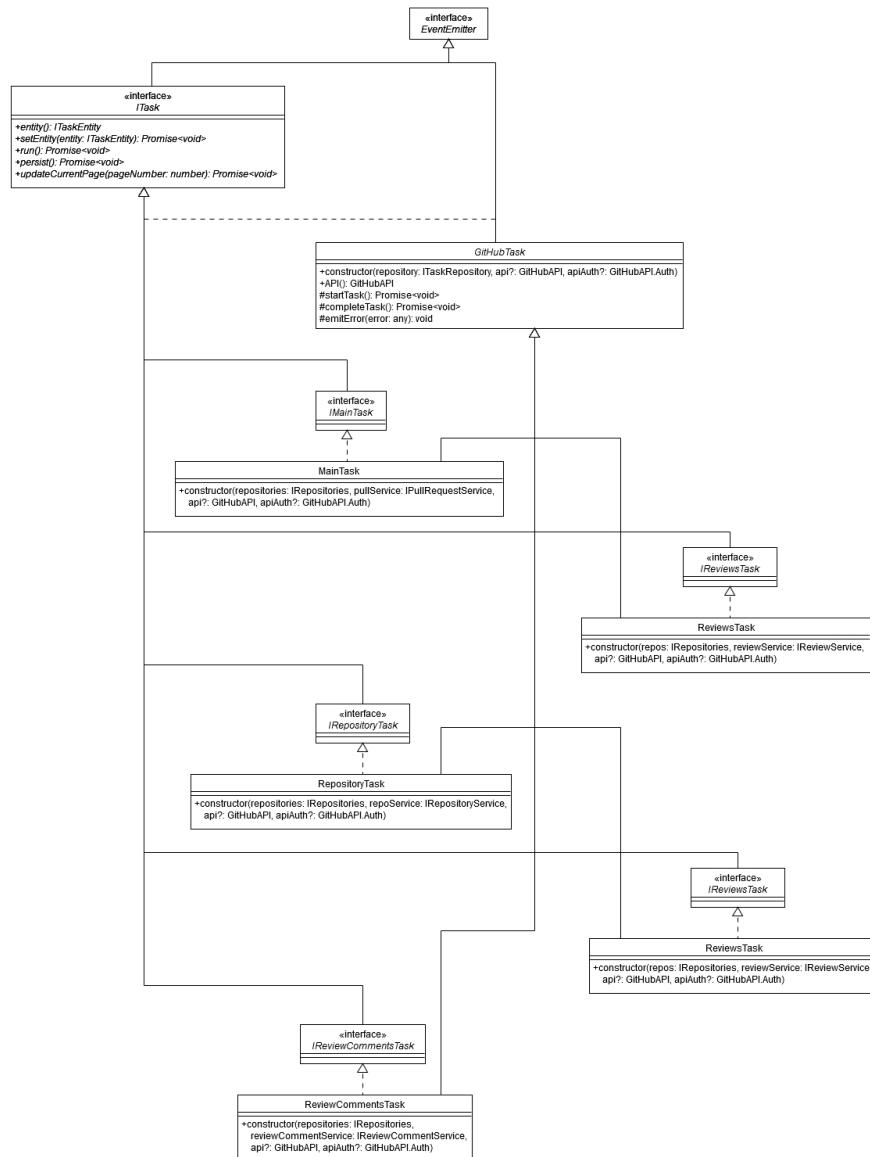


Figura C.7: Diagrama de clases de tareas (2).

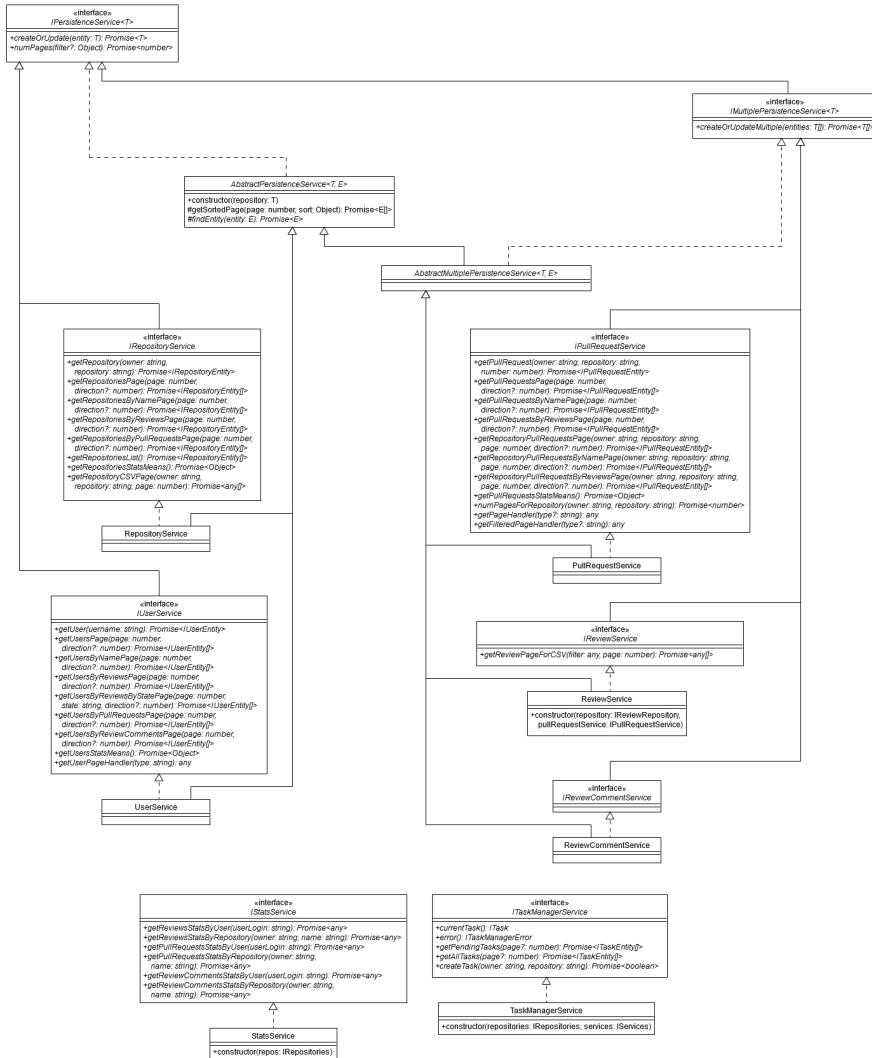


Figura C.8: Diagrama de clases de servicios.

## Diseño procedimental

Mediante el uso de diagramas de secuencia, se detallan los aspectos más relevantes del funcionamiento de la aplicación.

El siguiente diagrama de secuencia muestra, de forma general, la interacción entre el cliente web y la API REST desarrollada en el proceso de obtención de datos para su visualización.

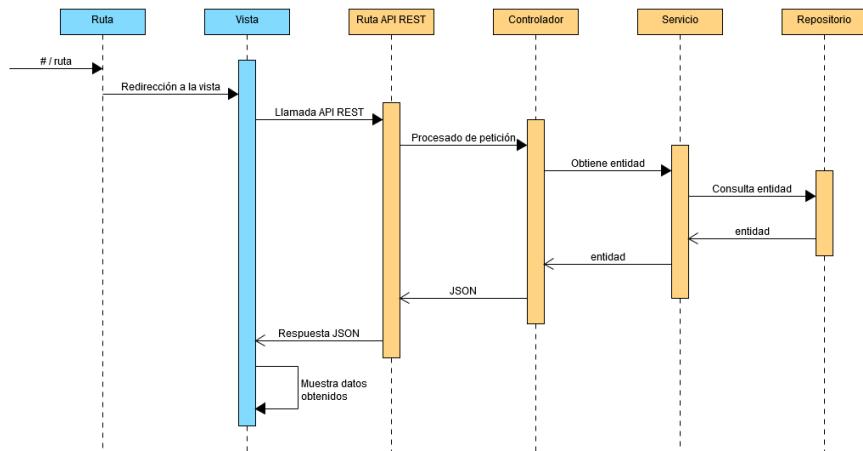


Figura C.9: Ejemplo de diagrama de secuencia cliente-API.

El diagrama de secuencia mostrado a continuación ilustra el proceso de ejecución de una tarea de obtención de datos de GitHub. En este caso necesita dos peticiones para completarse, pero con la primera petición supera el límite y la segunda retorna un error de límite superado. Se puede observar el funcionamiento del gestor de tareas ante esa situación.

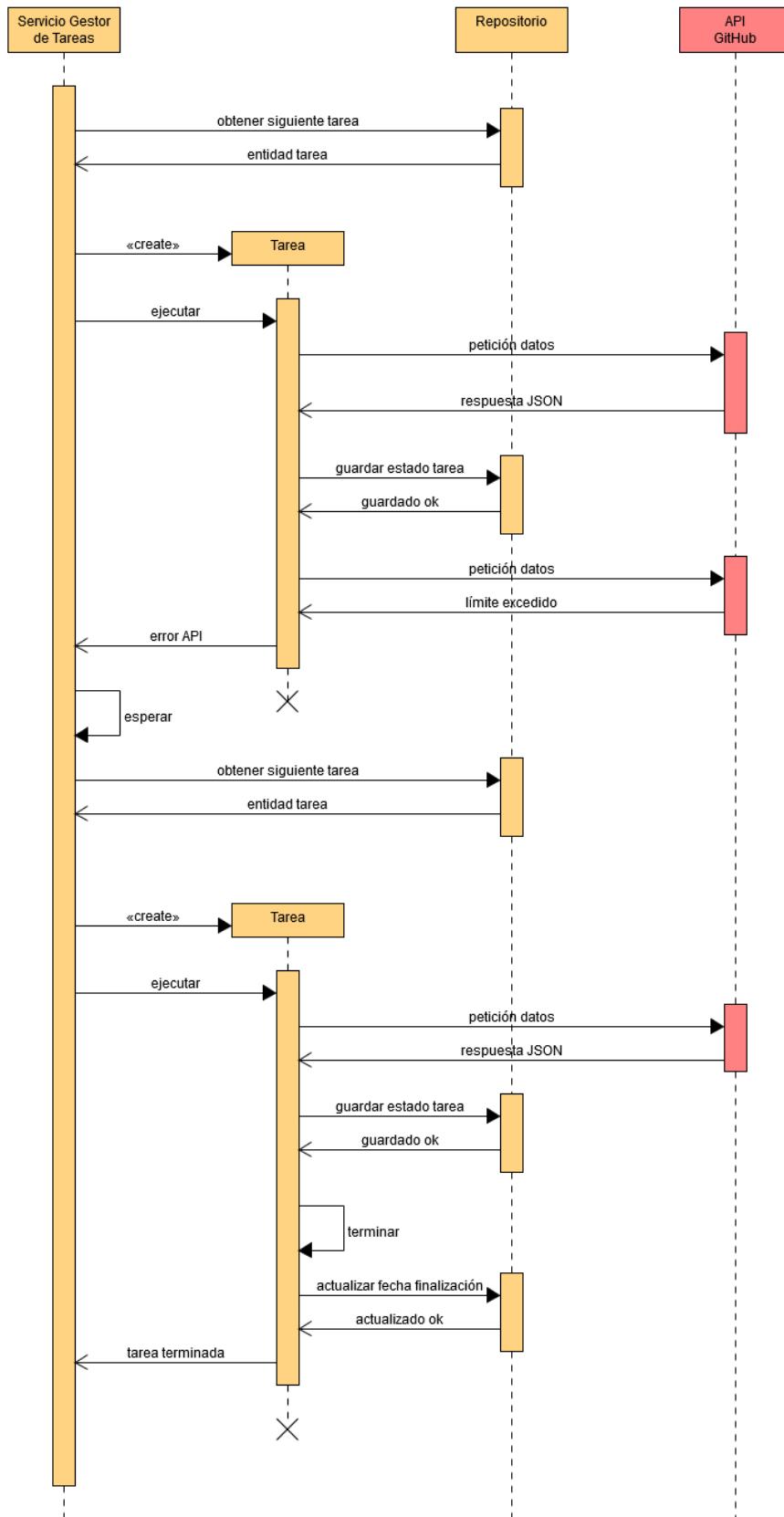


Figura C.10: Ejemplo de diagrama de secuencia de ejecución de una tarea.

## Diseño de interfaces

En esta parte se muestran los prototipos de las diferentes interfaces de la aplicación web.

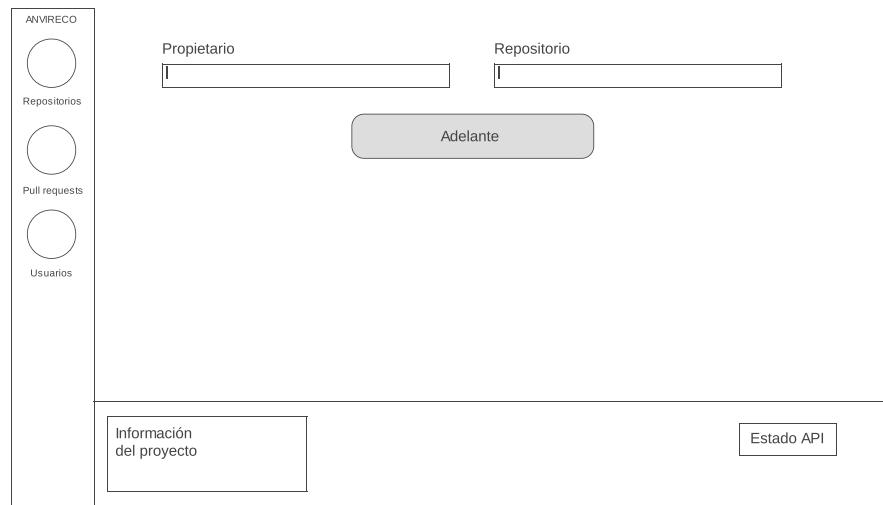


Figura C.11: Pantalla de inicio.

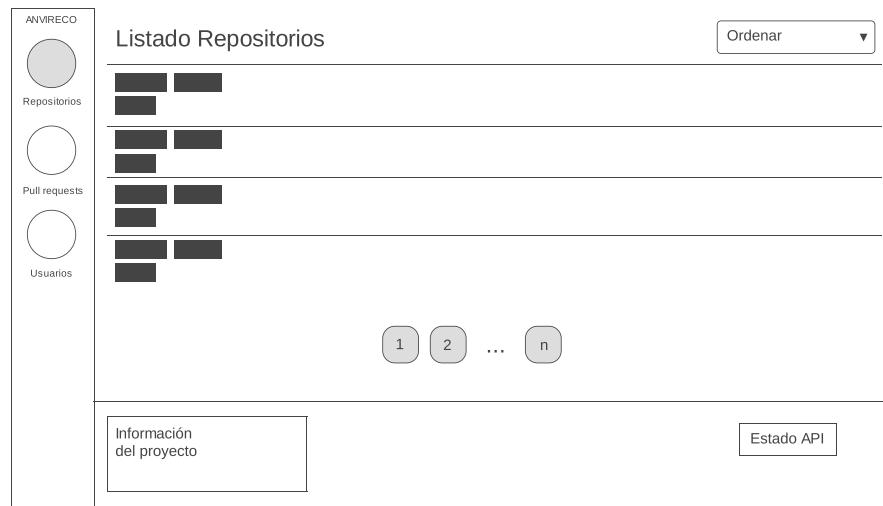


Figura C.12: Pantalla de listado de repositorios.

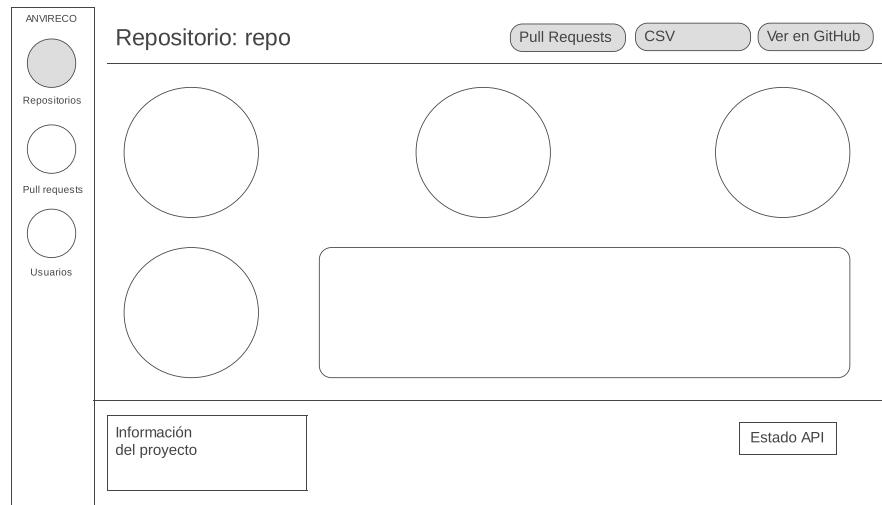
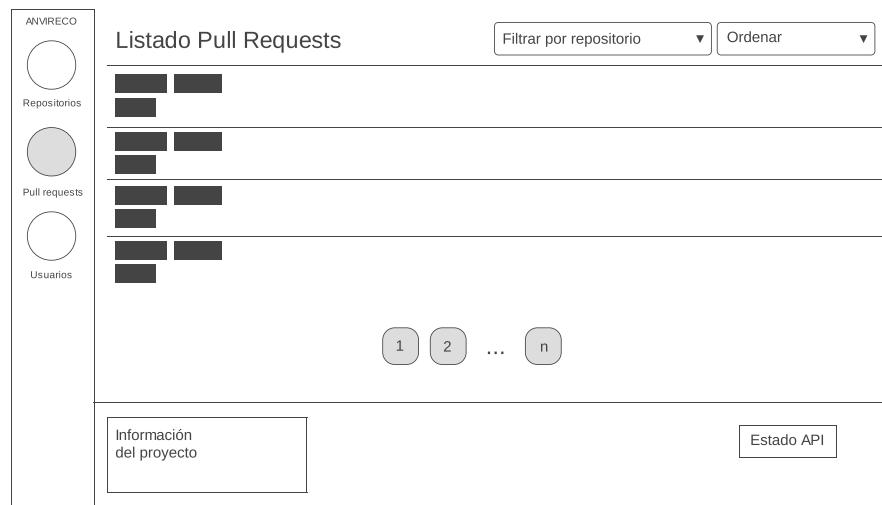


Figura C.13: Pantalla de análisis visual de repositorio.

Figura C.14: Pantalla de listado de *pull requests*.

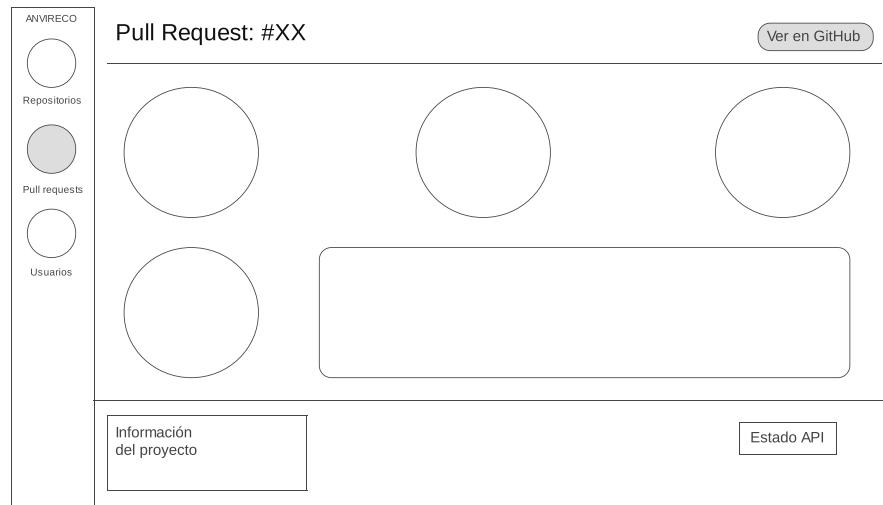
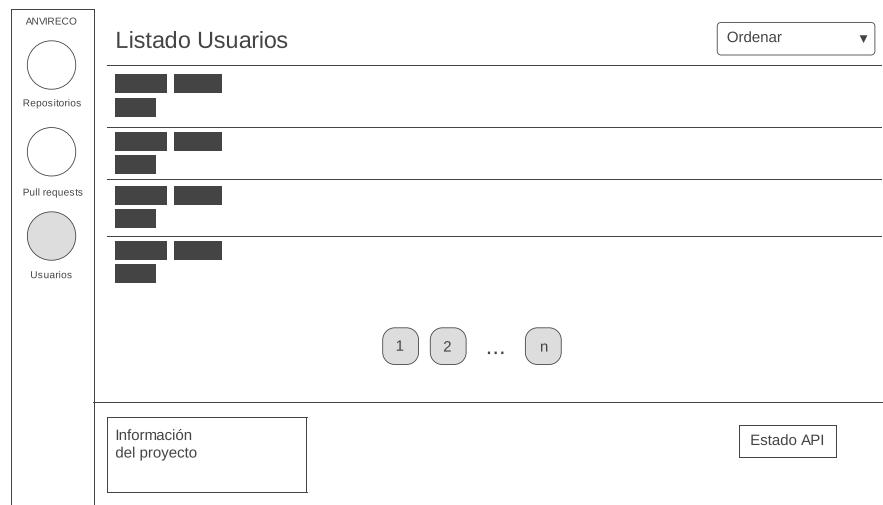
Figura C.15: Pantalla de análisis visual de *pull request*.

Figura C.16: Pantalla de listado de usuarios.

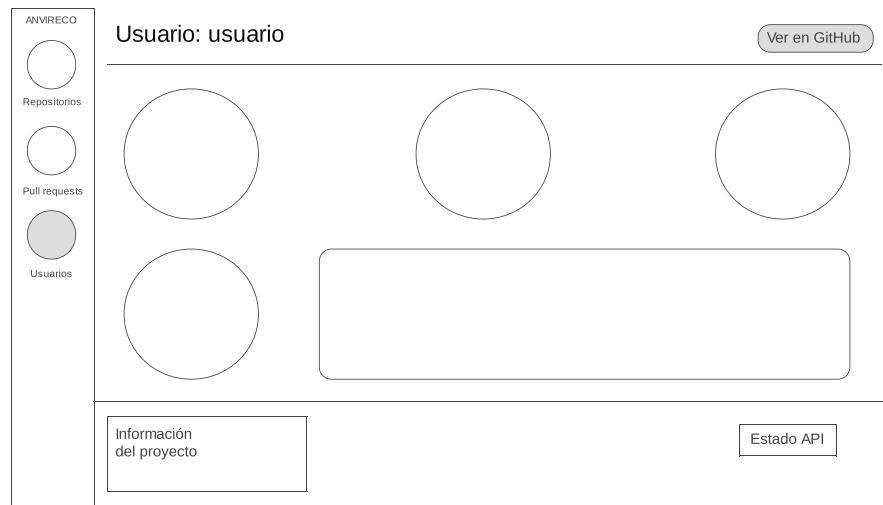


Figura C.17: Pantalla de análisis visual de usuario.

## Anexo D

# Documentación técnica de programación

### D.1. Introducción

Este anexo está dedicado al desarrollo de la documentación técnica de programación. Se tratan aspectos como la configuración del entorno de desarrollo, compilación y ejecución del proyecto, o pruebas realizadas.

### D.2. Estructura de directorios

El proyecto tiene la siguiente estructura de directorios:

- **/**: la raíz del proyecto, contiene ficheros de configuración de TypeScript, Travis, y Gulp, el fichero `package.json` de node.js, el fichero README, la licencia y ficheros para ignorar archivos en Git y en Codebeat.
- **/doc**: contiene toda la documentación del proyecto.
- **/doc/code**: documentación de código fuente.
- **/doc/memo**: memoria y anexos del proyecto.
- **/doc/memo/img**: figuras de la memoria y los anexos.
- **/doc/memo/tex**: ficheros `.tex`.
- **/node\_modules**: dependencias externas del proyecto.
- **/prototypes**: prototipos previos para la prueba de herramientas.
- **/prototypes/heroku**: un prototipo para probar el servicio Heroku.

- **/prototypes/nodejs:** un prototipo para probar node.js.
- **/release:** en esta ruta se copian los ficheros generados automáticamente tras la compilación. Contiene la aplicación lista para su ejecución.
- **/src:** código fuente del proyecto. Contiene las clases que crean el servidor web.
- **/src/app:** lógica de negocio y acceso a datos.
- **/src/app/data:** acceso a datos (repositorios).
- **/src/app/data/filters:** filtros para consultas MongoDB.
- **/src/app/data/schemas:** definición de schemas de MongoDB.
- **/src/app/entities:** entidades.
- **/src/app/entities/documents:** documentos *mongoose*.
- **/src/app/entities/enum:** enumeraciones.
- **/src/app/services:** lógica de negocio (servicios).
- **/src/app/services/tasks:** tareas de obtención de datos.
- **/src/app/util:** clases de utilidad.
- **/src/controllers:** controladores de la API REST.
- **/src/routes:** rutas de la API REST.
- **/test:** pruebas unitarias del proyecto.
- **/test/app:** pruebas unitarias de lógica de negocio y acceso a datos.
- **/test/app/data:** pruebas unitarias de acceso a datos (repositorios).
- **/test/app/entities:** pruebas unitarias de entidades.
- **/test/app/services:** pruebas unitarias de lógica de negocio (servicios).
- **/test/app/util:** pruebas unitarias de clases de utilidad.

### D.3. Manual del programador

El entorno de desarrollo debe satisfacer los siguientes requisitos mínimos:

- **Node.js y npm:** la aplicación está desarrollada en Node.js y hace uso del gestor de paquetes npm.
- **MongoDB:** la aplicación está preparada para trabajar con bases de datos NoSQL MongoDB. No es imprescindible tener una instancia local de MongoDB, como alternativa se pueden utilizar servicios en la nube como MongoDB Atlas o mLab.
- **Git:** la aplicación utiliza el control de versiones Git. Debe estar instalado en el sistema para la descarga del código fuente, y la publicación de cambios.

De forma opcional, pero recomendable se debería contar con:

- **GitHub Developer Application:** permite aumentar el límite de peticiones/hora a la API de GitHub de 60 a 5000.
- **Cliente Travis CI:** permite, entre otras cosas, encriptar datos en el fichero de configuración de Travis. Requiere Ruby instalado.

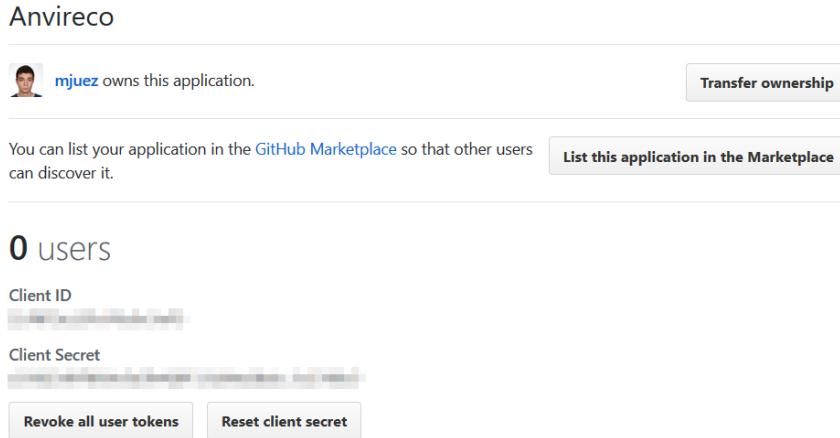


Figura D.1: GitHub Developer Application.

También es necesario un editor de código. Por la experiencia de uso en el desarrollo del proyecto, se recomienda Visual Studio Code [5], que integra herramientas para trabajar con TypeScript y con Git.

```

ReviewService.ts — git — Visual Studio Code
Archivo Editar Selección Ver Ir Depurar Ayuda
EXPLORADOR
EDTORES ABIERTOS
GIT
TS App.ts TS TaskManagerService.ts TS ReviewService.ts
1 import { IMultiplePersistenceService } from "../services/IPersistenceService";
2 import { IPullRequestService } from "../services/PullRequestService";
3 import { IReviewEntity } from "../entities/ReviewEntity";
4 import { IPullRequestEntity } from "../entities/PullRequestEntity";
5 import { IReviewRepository } from "../data/ReviewRepository";
6 import { AbstractMultiplePersistenceService } from "./AbstractPersistenceService";
7
8 /**
9  * Review service interface.
10 * Describes services for getting reviews data.
11 *
12 * @author Mario Juez <mario[at]mjuez.com>
13 */
14 export interface IReviewService extends IMultiplePersistenceService<IReviewEntity> {
15
16   /**
17    * Gets a page of data of filtered reviews.
18    * The data is prepared to be easily converted to CSV.
19    *
20    * @param filter reviews filter (usually by repository).
21    * @param page page number.
22    * @returns an array of reviews data.
23   */
24   getReviewPageForCSV(filter: any, page: number): Promise<any[]>;
25
26 }
27
28 /**
29  * Review services implementation.
30 */

```

Lin. 45, Col. 87 (70 seleccionada) Espacios: 4 UTF-8 CR LF TypeScript 2.3.4

Figura D.2: Visual Studio Code.

## Contenedor de aplicación Heroku

Actualmente la aplicación se despliega en Heroku. Para desplegar la aplicación en este servicio, en primer lugar se debe crear una cuenta de usuario. Tras ello, en el panel de administración, se debe crear una aplicación.

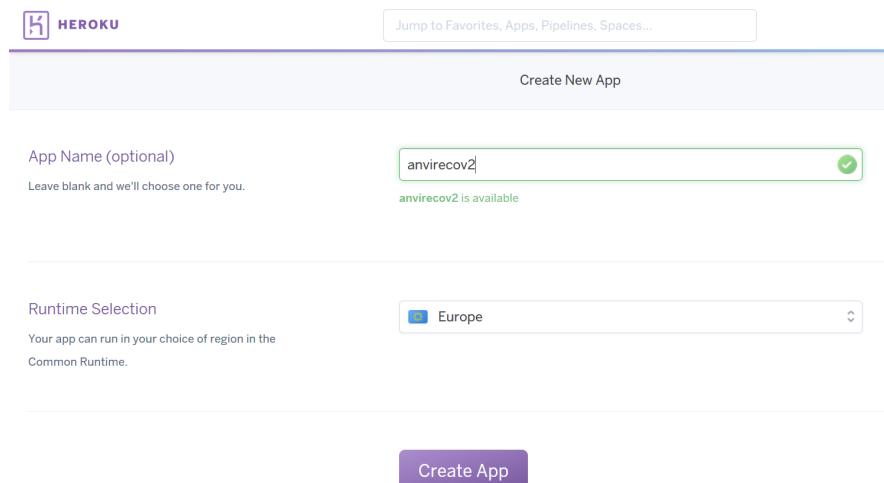


Figura D.3: Creando una aplicación en Heroku.

Una vez creada la aplicación, se debe seleccionar lo que Heroku denomina

*buildpack*, concretamente el *buildpack* de node.js. Gracias a esto, Heroku configura el modo de construir y ejecutar la aplicación de forma automática. Esta configuración se puede encontrar en la pestaña de ajustes de la aplicación.

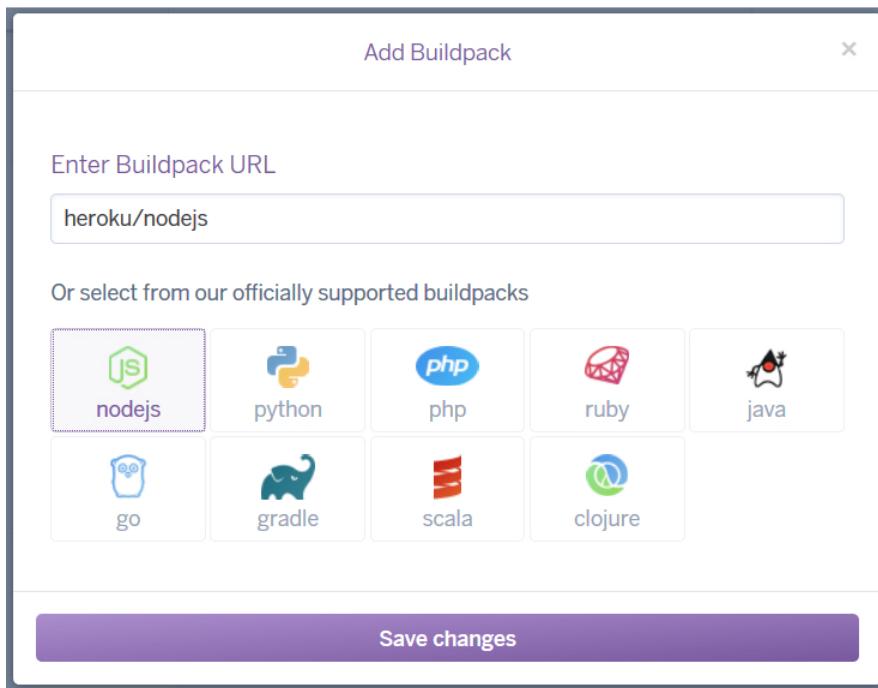


Figura D.4: Seleccionando un *buildpack* en Heroku.

Como la aplicación se despliega a través de Travis, en Heroku no hay que hacer ninguna configuración más.

### Integración continua con Travis

Gracias al uso de Travis, cada vez que se publican cambios en una de las ramas `dev` o `master`, esos cambios se despliegan automáticamente en su aplicación correspondiente en Heroku.

En primer lugar es necesario obtener una cuenta de Travis. Para ello, la herramienta da la opción de utilizar el usuario de GitHub.

El siguiente paso consiste en habilitar el uso de Travis en el repositorio:



Figura D.5: Conectando el repositorio con Travis.

Finalmente se debe añadir un fichero de configuración `.travis.yml` en la raíz del repositorio.

```

1 language: node_js
2 node_js:
3   - '7'
4 branches:
5   only:
6     - master
7     - dev
8 install:
9   - npm install
10 script:
11   - npm test
12 deploy:
13   provider: heroku
14   api_key:
15     secure: o8V/yYy59uVMYoxaIUBoS
16   app:
17     master: anvireco
18     dev: anvireco-dev

```

Figura D.6: Configuración de `.travis.yml`.

En este fichero se detalla lo siguiente:

1. En las líneas 1-3 se indica a Travis que el lenguaje utilizado es node.js en su versión 7.
2. En las líneas 4-7 se indica a Travis que únicamente se debe ejecutar al actualizar las ramas `master` o `dev`. Con cualquier otra rama, por ejemplo `memo`, Travis no se lanzará.

3. En las líneas 8-9 se indica a Travis qué comando debe ejecutar para instalar la aplicación. En este caso `npm install` descarga las dependencias y compila la aplicación.
4. En las líneas 10-11 se indica a Travis qué comandos adicionales debe ejecutar. En este caso `npm test`, para pasar las pruebas unitarias.
5. En las líneas 12-13 se indica a Travis que debe desplegar la aplicación, concretamente en Heroku.
6. En las líneas 14-15 se añade la *APK-KEY* de Heroku (cada usuario tiene una *API-KEY*). Dicha clave se añade al fichero a través del cliente de Travis para encriptarla, con el siguiente comando: `travis encrypt $(API-KEY) --add deploy.api.key`.
7. En las líneas 16-18 se indica a Travis en qué aplicación de Heroku deben ser desplegados los cambios de una rama concreta. La rama `master` se despliega en la aplicación `avireco`, y la rama `dev` en `avireco-dev`.

Una vez configurado Travis, el proceso de construcción, pruebas y despliegue queda completamente automatizado. Desde GitHub y desde la web de Travis es posible ver el estado de cada construcción.

```

mjuez / TFM2016_Analisis-Visual-Revisões-Código build passed

Current Branches Build History Pull Requests More options ⚙

✓ master Merge branch 'dev' #172 passed Restart build

Commit 3366b3e Ran for 2 min 5 sec  

Compare 6914ab8...3366b3e a day ago  

Branch master  

Mario Juez authored and committed

Job log View config
Remove log Raw log
Worker information  
Build systems information  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
Setting environment variable from .travis.yml  
$ export DEBIAN_FRONTEND=noninteractive  
$ sudo apt-get install libcurl3  
$ git clone --depth=50 --branch=master https://github.com/mjuez/TFM2016_Analisis-Visual-Revisões-Código.git  
fix.CVE-2015-1547  
update libcurl 0.10.0  
git checkout 3.40.0

```

Figura D.7: Ejemplo de estado de construcción en Travis.

### Revisiones automáticas de código con Codebeat

Mediante el uso de Codebeat se realizan revisiones automáticas de código cada vez que se realizan cambios en la rama `master` del proyecto o cada vez que se añaden cambios a una *pull request* abierta para integrar cambios en la rama `master`.

Para utilizarlo es necesario obtener una cuenta de Codebeat. Para ello, la herramienta da la opción de utilizar el usuario de GitHub.

A continuación se debe escoger el repositorio para el cual se desean realizar revisiones automáticas de código:

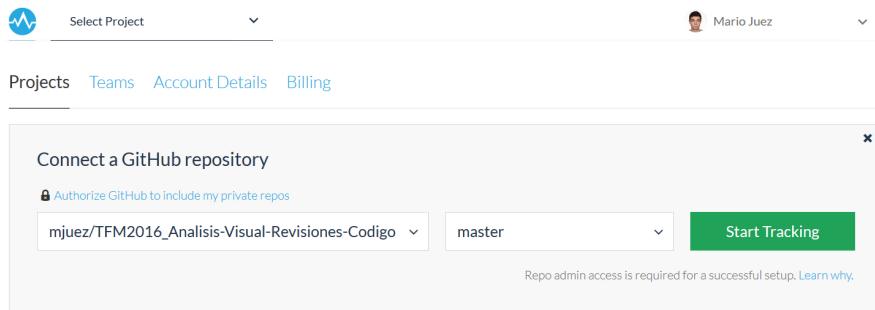


Figura D.8: Selección de repositorio para realizar revisiones con Codebeat.

Tras ello, la herramienta ya está preparada para realizar revisiones de forma automática. Éstas funcionan como *checks* de GitHub.



Figura D.9: Codebeat como *check* de GitHub.

## Generación de la documentación de código

La documentación de código se genera con la herramienta TypeDoc. Si situados en la raíz del proyecto, mediante el siguiente comando se guarda la documentación en la carpeta `/doc/code`:

---

```
typedoc --out doc/code/ . --exclude prototypes/
```

---

Con `--out` se indica el directorio de destino, a continuación se debe indicar el directorio donde se encuentra el proyecto, como estamos situados en la raíz, será el directorio actual (`.`). Con `--exclude` se indica que directorios deben

ser excluidos en el proceso de generación de documentación, en este caso la carpeta de prototipos.

## D.4. Compilación, instalación y ejecución del proyecto

En primer lugar se debe descargar el código fuente de la aplicación alojado en GitHub. Bien el fichero .zip generado por GitHub, o clonando el repositorio:

---

```
git clone https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo.git
```

---

Una vez descargada y descomprimida, es necesario situarse dentro de la carpeta que contiene el fichero package.json.

### Configuración de variables de entorno

Las variables de configuración de la aplicación se deben definir como variables de entorno. Dependiendo del sistema operativo utilizado, la forma de declarar variables de entorno varía.

Se deben declarar las siguientes variables de entorno:

- **MONGO\_CONNSTRING:** cadena de conexión a la base de datos MongoDB (formato según [6]).
  - **Ejemplo:** mongodb://admin:1234@127.0.0.1:27017/anvireco
- **ANVIRECO\_APPNAME:** nombre de una aplicación GitHub Developer Application [3].
  - **Ejemplo:** Anvireco
- **ANVIRECO\_ID (opcional):** *client ID* de una aplicación GitHub Developer Application [3].
  - **Ejemplo:** 00x000x00xxx0x0x0xx0
- **ANVIRECO\_SECRET (opcional):** *client secret* de una aplicación GitHub Developer Application [3].
  - **Ejemplo:** 00x000x00xxx0x0xx000x000x00xxx0x0xx0
- **PORT (opcional):** Puerto donde se va a ejecutar el servidor node.js.
  - **Ejemplo:** 3000

Por ejemplo, las variables de entorno en Heroku se configuran en la pestaña de ajustes de la aplicación.

The screenshot shows the Heroku Settings page for the app 'anvireco'. At the top, there are tabs for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings, with 'Settings' being the active tab. Below the tabs, there's a section for 'Config Variables' with a note: 'Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.' A table lists five config variables:

Config Vars	Value	Edit	Delete
ANVIRECO_APPNAME	[REDACTED]		
ANVIRECO_ID	[REDACTED]		
ANVIRECO_SECRET	[REDACTED]		
MONGO_CONNSTRING	[REDACTED]		
KEY	VALUE	Add	

Figura D.10: Variables de entorno en Heroku.

## Instalación de dependencias

Existen dos formas de instalar las dependencias de la aplicación utilizando el gestor de paquetes `npm`:

- **Entorno de producción:** instala únicamente las dependencias imprescindibles para la ejecución de la aplicación.

---

```
npm install --only=production
```

---

- **Entorno de desarrollo:** instala todas las dependencias, incluyendo aquellas opcionales como por ejemplo las de *testing*.

---

```
npm install
```

---

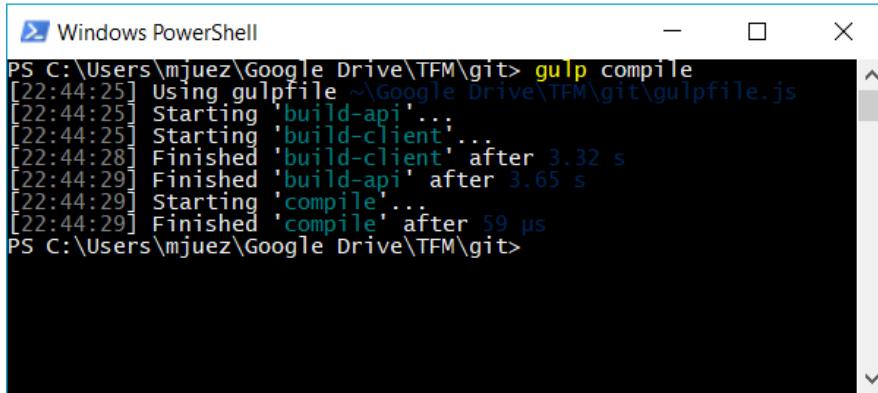
## Compilación de ficheros TypeScript

Los ficheros TypeScript (`.ts`) dentro del directorio `src` son compilados a JavaScript (`.js`) y guardados en el directorio `release`. La compilación se realiza mediante un objetivo de Gulp llamado `compile`.

---

```
gulp compile
```

---



```
PS C:\Users\mjuez\Google Drive\TFM\git> gulp compile
[22:44:25] Using gulpfile ~\Google Drive\TFM\git\gulpfile.js
[22:44:25] Starting 'build-api'...
[22:44:25] Starting 'build-client'...
[22:44:28] Finished 'build-client' after 3.32 s
[22:44:29] Finished 'build-api' after 3.65 s
[22:44:29] Starting 'compile'...
[22:44:29] Finished 'compile' after 59 µs
PS C:\Users\mjuez\Google Drive\TFM\git>
```

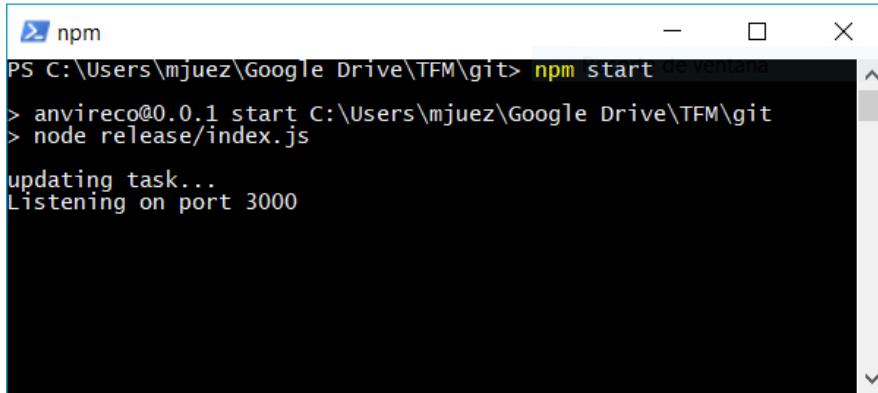
Figura D.11: Compilación de la aplicación.

### Ejecución de la aplicación

La aplicación se ejecuta mediante el comando `npm start`. La aplicación es accesible desde `http://localhost:puerto` donde `puerto` es el valor de la variable de entorno `PORT`.

Si no se ha definido la variable de entorno `PORT`, la aplicación se desplegará por defecto en el puerto 3000 y será accesible a través de `http://localhost:3000/`.

Para la ejecución de la aplicación es imprescindible la compilación previa de los ficheros TypeScript.



```
PS C:\Users\mjuez\Google Drive\TFM\git> npm start
> anvireco@0.0.1 start C:\Users\mjuez\Google Drive\TFM\git
> node release/index.js
updating task...
Listening on port 3000
```

Figura D.12: Ejecución de la aplicación.

## D.5. Pruebas del sistema

Las pruebas del sistema son pruebas unitarias desarrolladas con *mocha*, *chai*, y *sinon*, las cuales son conocidas librerías de *testing* para proyectos en node.js.

La ejecución de las pruebas unitarias solo se puede llevar a cabo si se han instalado todas las dependencias. Para la ejecución de las pruebas no es necesario compilar los ficheros TypeScript.

Mediante el comando `npm test` se ejecutan las pruebas.

El proyecto tiene un conjunto de 18 pruebas unitarias dedicadas principalmente a la prueba de las clases relacionadas con las *pull requests*.

```
anvireco@0.0.1 test C:\Users\mjuez\Google Drive\TFM\git
mocha --reporter spec --compilers ts:ts-node/register test/**/*.test.ts

Checking Pull Request repository
✓ Should return created (persisted) entity
✓ Should get the mongoose Model
✓ Should return the number of updated entities
✓ Should return an array with all entities
✓ Should return one pull request (found one)
✓ Should return null for a pull request id that not exists

Checking Pull Request entities
✓ Should transform an object to a IPullRequestEntity
✓ The IPullRequestEntity document should contain the data of the source object
✓ Should transform an array of objects to a IPullRequestEntity array
✓ Should get its id
✓ Should get its entire document

Checking Pull Request services
✓ Should create pull request into database
✓ Should update pull request into database
✓ Should create or update many pull requests

Checking GitHub utilities
✓ Next page should be 3
✓ No next page should be found
✓ Repository should exist (1110ms)
✓ Repository should not exist (1668ms)

18 passing (3s)
```

Figura D.13: Pruebas unitarias del sistema.

## Anexo E

# Documentación de usuario

### E.1. Introducción

Este anexo contiene la documentación necesaria para que el usuario conozca y aprenda el funcionamiento de la aplicación.

El usuario siempre accederá a la última versión de la aplicación debido a que ésta se encuentra desplegada en un servidor remoto y no precisa de instalación local. Se sigue un proceso de desarrollo incremental, lo que supone que la funcionalidad de la aplicación está en constante evolución. Por ello la documentación de usuario debe ser actualizada periódicamente.

Para que el usuario siempre tenga a su disposición la última versión de la documentación de usuario, ésta se encuentra accesible desde la *Wiki* del repositorio.

### E.2. Requisitos

La aplicación desarrollada se despliega en la nube, lo que quiere decir que no necesita instalación.

Para el uso de la API REST, el usuario deberá tener instalado cualquier software que permita realizar peticiones y recibir respuestas HTTP. Cualquier navegador web permite este tipo de operaciones, aunque también existen aplicaciones desarrolladas específicamente para trabajar con APIs como por ejemplo Postman.

Para el uso del cliente web, será necesario utilizar la última versión de uno de los tres navegadores principales:

- Microsoft Edge
- Mozilla Firefox

- Google Chrome

También es imprescindible que el navegador tenga el uso de JavaScript habilitado. La aplicación debería funcionar correctamente en otros navegadores (por ejemplo Safari), pero solo se ha probado en los tres indicados anteriormente.

### E.3. Manual del usuario

Los manuales de usuario se pueden encontrar en la *Wiki* del repositorio, en el siguiente enlace:

---

[https://github.com/mjuez/TFM2016\\_Analisis-Visual-Revisiones-Codigo/wiki](https://github.com/mjuez/TFM2016_Analisis-Visual-Revisiones-Codigo/wiki)

---

The screenshot shows a GitHub repository page for 'mjuez / TFM2016\_Analisis-Visual-Revisiones-Codigo'. The top navigation bar includes links for Features, Business, Explore, Marketplace, Pricing, and a sign-in/sign-up button. The repository name is 'mjuez / TFM2016\_Analisis-Visual-Revisiones-Codigo'. Below the name, there are buttons for Watch (4), Star (0), and Fork (0). The main navigation tabs are 'Code', 'Issues (4)', 'Pull requests (0)', 'Wiki' (which is selected and highlighted in orange), and 'Insights'. The 'Home' section contains a brief description of the tool: 'Anvireco es una herramienta para la extracción y visualización de datos sobre revisiones de código realizadas en repositorios de código alojados en GitHub.' It also mentions that Mario Juez edited the page 15 days ago with 5 revisions. To the right of the home content is a sidebar with a table of contents for the wiki pages, including sections like 'Inicio', 'Uso de la API', 'Uso del cliente', and 'Obtención de datos'. At the bottom right of the sidebar is a link 'Clone this wiki locally'.

Figura E.1: Fragmento de la *Wiki* en GitHub.

# Bibliografía

- [1] Creative Commons. Attribution 4.0 international (cc by 4.0). <https://creativecommons.org/licenses/by/4.0/>. [Internet; accedido 28-junio-2017].
- [2] Ministerio de Empleo y Seguridad Social. Bases y tipos de cotización 2017. [http://www.seg-social.es/Internet\\_1/Trabajadores/CotizacionRecaudacion10777/Basesytiposdecotiza36537/index.htm](http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudacion10777/Basesytiposdecotiza36537/index.htm). [Internet; accedido 27-junio-2017].
- [3] GitHub. Developer applications. <https://github.com/settings/developers>. [Internet; accedido 29-junio-2017].
- [4] Open Source Initiative. Mit license. <https://opensource.org/licenses/MIT>. [Internet; accedido 28-junio-2017].
- [5] Microsoft. Visual studio code. <https://code.visualstudio.com/>. [Internet; accedido 29-junio-2017].
- [6] MongoDB. Connection string uri format. <https://docs.mongodb.com/manual/reference/connection-string/>. [Internet; accedido 29-junio-2017].