

# Sistema centralizado de autenticación para entorno empresarial

Daniel Ignacio Salazar Recio

Proyecto presentado para la titulación de  
Ingeniería Informática

Supervisado por:  
Manuel Palomo Duarte



Universidad de Cádiz  
Septiembre de 2017

*Yo, Daniel Ignacio Salazar Recio confirmo que el trabajo presentado en este proyecto está hecho por mí. Cuando la información viene derivada de otras fuentes, confirmo que se indice explícitamente en la memoria.*

# Resumen

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam et turpis gravida, lacinia ante sit amet, sollicitudin erat. Aliquam efficitur vehicula leo sed condimentum. Phasellus lobortis eros vitae rutrum egestas. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec at urna imperdiet, vulputate orci eu, sollicitudin leo. Donec nec dui sagittis, malesuada erat eget, vulputate tellus. Nam ullamcorper efficitur iaculis. Mauris eu vehicula nibh. In lectus turpis, tempor at felis a, egestas fermentum massa.

# Agradecimientos

Me gustaria dar las gracias a mi familia por haberme apoyado durante el desarrollo del proyecto en especial mis padres y mi hermano, así como a los compañeros de la carrera que me han ayudado con consejos y ánimos. También dar las gracias a mi tutor de proyecto Manuel Palomo por los consejos y supervisión.

# Table of Contents

<b>Resumen</b>	<b>i</b>
<b>Agradecimientos</b>	<b>ii</b>
<b>Lista de figuras</b>	<b>iii</b>
<b>Lista de tablas</b>	<b>iv</b>
<b>Definiciones y acrónimos</b>	<b>v</b>
<b>1 Introducción</b>	<b>1</b>
1.1 Objetivos y alcance . . . . .	1
1.2 Estructura del documento . . . . .	1
<b>2 Planificación</b>	<b>3</b>
2.1 Fase inicial . . . . .	3
2.2 Fase de análisis . . . . .	3
2.3 Fase de aprendizaje . . . . .	4
2.4 Fase de diseño . . . . .	4
2.5 Implementación . . . . .	4
2.6 Pruebas . . . . .	4
2.7 Redacción de la memoria . . . . .	5
2.8 Diagrama de Gantt . . . . .	5
<b>3 Descripción general del proyecto</b>	<b>6</b>
3.1 Descripción . . . . .	6
3.2 Perfiles de usuario . . . . .	6
3.2.1 Perfil Administrador . . . . .	7
3.2.2 Perfil Gestor de aplicaciones . . . . .	7

3.2.3	Perfil Usuario . . . . .	7
3.3	Interfaz de Usuario . . . . .	7
3.4	Software . . . . .	7
<b>4</b>	<b>Análisis</b>	<b>9</b>
4.1	Metodología de desarrollo . . . . .	9
4.2	Especificación de requisitos del sistema . . . . .	9
4.2.1	Gestión de instalación . . . . .	9
4.2.1.1	Caso de uso: Finalizar instalación . . . . .	10
4.2.2	Gestión de usuarios . . . . .	11
4.2.2.1	Caso de uso: Añadir usuario . . . . .	11
4.2.2.2	Caso de uso: Terminar configuración de usuario . . . . .	12
4.2.2.3	Caso de uso: Editar usuario . . . . .	12
4.2.3	Gestión de empresas . . . . .	13
4.2.3.1	Caso de uso: Añadir empresa . . . . .	13
4.2.3.2	Caso de uso: Editar empresa . . . . .	14
4.2.3.3	Caso de uso: Ver empresa. . . . .	15
4.2.4	Gestión de productos . . . . .	15
4.2.4.1	Caso de uso: Seleccionar producto . . . . .	15
4.2.4.2	Caso de uso: Registrar aplicación . . . . .	16
4.2.4.3	Caso de uso: Editar aplicación . . . . .	16
4.2.4.4	Caso de uso: Borrar aplicación . . . . .	17
4.2.4.5	Caso de uso: Ver detalle de aplicación . . . . .	18
4.2.4.6	Caso de uso: Editar usuarios de la empresa con acceso a la aplicación . . . . .	18
4.3	Integraciones . . . . .	19
4.3.0.1	Caso de uso: Registrar aplicación externa . . . . .	19
4.3.0.2	Caso de uso: Obtener token de usuario desde aplicación interna . . . . .	20
4.3.0.3	Caso de uso: Obtener token de usuario desde aplicación externa . . . . .	21
4.3.0.4	Caso de uso: Importar usuarios desde mó- dulo externo . . . . .	21

4.3.0.5	Caso de uso: Sincronizar usuarios con módulo externo . . . . .	22
<b>5</b>	<b>Diseño</b>	<b>26</b>
5.1	Introducción . . . . .	26
5.2	Arquitectura de aplicación . . . . .	27
5.2.1	Elección del framework . . . . .	27
5.2.2	Endpoints . . . . .	27
5.3	Base de datos . . . . .	27
5.3.1	Modelo entidad-relación . . . . .	28
5.3.2	Tablas y atributos . . . . .	28
5.4	Arquitectura del sistema . . . . .	28
5.4.1	API . . . . .	28
5.4.2	Base de datos . . . . .	29
5.4.3	Worker . . . . .	29
5.4.4	Servidor de aplicaciones . . . . .	29
5.4.5	Servidor web . . . . .	30
5.4.6	Infraestructura . . . . .	30
5.4.7	Escalabilidad . . . . .	31
5.4.7.1	Balancedor de carga . . . . .	31
5.4.7.2	Servicios autoescalables . . . . .	32
5.4.7.3	Alta disponibilidad . . . . .	32
5.5	Despliegue . . . . .	32
5.5.1	Despliegue de software . . . . .	33
5.5.1.1	Paquetizado . . . . .	33
5.5.1.2	Gestión de configuración . . . . .	33
5.5.2	Gestión de infraestructura . . . . .	34
5.6	Métricas y monitorización . . . . .	34
5.6.1	Métricas . . . . .	34
5.6.2	Monitorización . . . . .	34
5.6.3	Alertas . . . . .	34
<b>6</b>	<b>Final research study</b>	<b>35</b>
6.1	Introduction . . . . .	35
6.2	Method . . . . .	35



6.2.1	Subsection 1 . . . . .	35
6.2.2	Subsection 2 . . . . .	36
6.3	Results . . . . .	36
6.4	Discussion . . . . .	36
6.5	Conclusion . . . . .	36
<b>7</b>	<b>Conclusion</b>	<b>37</b>
7.1	Thesis summary . . . . .	37
7.2	Future work . . . . .	37
	<b>Appendix 1: Some extra stuff</b>	<b>38</b>
	<b>Appendix 2: Some more extra stuff</b>	<b>39</b>
<b>8</b>	<b>References</b>	<b>40</b>

# Lista de figuras

Figure 4.1 This is an example figure . . .	pp
Figure x.x Short title of the figure . . .	pp

# Lista de tablas

Table 5.1 This is an example table . . .	pp
Table x.x Short title of the figure . . .	pp

# Definiciones y acrónimos

<b>API</b>	<b>A</b> pplication <b>P</b> rogramming <b>I</b> nterface
<b>JSON</b>	<b>J</b> ava <b>S</b> cript <b>O</b> bject <b>N</b> otation
<b>Python</b>	Lenguaje interpretado del lado del servidor.
<b>HTML</b>	<b>H</b> yper <b>T</b> ext <b>M</b> arkup <b>L</b> anguage

# Chapter 1

## Introducción

Con este Proyecto de Fin de Carrera se pretende la consecución de dos objetivos fundamentales: poner en práctica los conocimientos adquiridos en la titulación de Ingeniería en Informática y desarrollar una aplicación que de solución a un problema real de un entorno empresarial.

### 1.1 Objetivos y alcance

El proyecto consiste en la creación de un software que centralice y homogenice la autenticación en un entorno empresarial. Una problemática habitual de una empresa que utilice múltiples productos es la diferencia entre los diferentes sistemas de autenticación de cada una de las aplicaciones, de forma que complica la gestión de estos usuarios debido a lo heterogéneo de los diferentes sistemas. Con este proyecto se propone un sistema que utiliza OAuth 2.0, de forma que sea sencillo de integrar con otras aplicaciones y facilitar también la gestión de permisos de cada una de las aplicaciones, centralizándolo en un único programa de gestión.

### 1.2 Estructura del documento

El documento se compone de los siguientes capítulos:

- Introducción: descripción del proyecto, objetivos y alcance del mismo y estructura básica del documento.
- Planificación: descripción del desarrollo de la planificación temporal y plazos de realización. Descripción general: descripción detallada sobre el proyecto, especificando tecnologías y herramientas usadas para su desarrollo.
- Análisis: fase de análisis del sistema, empleando la metodología seleccionada. Definición de requisitos funcionales del sistema, modelo conceptual y modelo de comportamiento.
- Diseño: diferentes fases del diseño técnico, arquitectura del sistema, diseño de la base de datos y diagramas de clase aplicadas al diseño.
- Implementación: aspectos más relevantes de la fase de implementación del sistema y explicación de los problemas encontrados durante el desarrollo.
- Pruebas y validaciones: plan de pruebas y automatización utilizado para el proyecto. Conclusiones: valoración y conclusiones personales obtenidas tras la realización del proyecto. Apéndices: \*\* Manual de instalación: manual para instalar correctamente la aplicación. \*\* Manual de usuario: manual para ayudar al usuario en el uso de la aplicación.
- Bibliografía: libros y referencias consultadas durante la realización del proyecto.
- Licencia GPL 3: texto completo sobre la licencia GPL 3, por la cual se rige el proyecto.

# Chapter 2

## Planificación

La planificación se divide en varias fases, a continuación se explicará en detalle cada una de ellas.

### 2.1 Fase inicial

La primera fase consistió en el planteamiento de la idea del proyecto, que en principio era desarrollar una aplicación que diera solución a un problema real del mundo empresarial. Tras la experiencia laboral adquirida en los últimos años decido realizar este proyecto conociendo los múltiples requisitos que pueden entrar en juego.

### 2.2 Fase de análisis

Se realizan diversas reuniones con empleados de una empresa para captar requisitos de manera informal. Tras estas reuniones se repasan estos requisitos y se escriben de manera formal, en forma de historias de usuario con tests de aceptación. En cada reunión posterior se plantean dudas y se van refinando esos requisitos. Finalmente se elabora un documento de requisitos que queda validado por los stakeholders del proyecto, en este caso los futuros usuarios de esta empresa.

## 2.3 Fase de aprendizaje

Para la realización del proyecto usaron tecnologías de las que se tenían conocimiento. La fase de aprendizaje consistió principalmente en entender la problemática de la empresa en detalle, entendiendo que tenía que ser una solución que se aplicara de forma genérica.

## 2.4 Fase de diseño

Fase en la que se realiza el diseño técnico de la arquitectura de la aplicación. Hay que entender como funcionará la aplicación a alto nivel, estructurarla de forma que sea escalable y que pueda atender peticiones de múltiples usuarios.

También entra en esta fase el diseño de la estructura interna de la aplicación, patrones de diseño utilizados, etc.

## 2.5 Implementación

Fase más extensa del desarrollo del proyecto. Consiste en implementar los requisitos especificados en la fase de análisis siguiendo para ello el diseño realizado en la fase anterior, procurando que la aplicación final satisfaga las necesidades y cumpla también los requisitos no funcionales.

## 2.6 Pruebas

Etapa importante en la que se comprueba una por una las funcionalidades del sistema verificando que no hay errores y que todo funciona como debe. Se realiza un plan de pruebas que pueda ser automatizado.





## Chapter 3

# Descripción general del proyecto

Este proyecto tiene la condición de Software Libre, por lo que en caso de necesitar ser ampliado, cualquier persona podría hacerlo. El proyecto es una aplicación nueva, no es continuación de otro proyecto

### 3.1 Descripción

El proyecto consiste en una aplicación web, con una API pública, con distintos perfiles de usuario, en la que se llevará a cabo la configuración de las aplicaciones que tienen acceso al sistema.

### 3.2 Perfiles de usuario

A continuación se expondrán los diferentes perfiles detallando a que funcionalidad tendrá acceso cada uno.

### 3.2.1 PERFIL ADMINISTRADOR

El administrador tendrá acceso a toda la gestión de usuarios, de productos y permisos. Por tanto el administrador podrá crear nuevos usuarios con los perfiles que considere necesarios, nuevas aplicaciones y otorgar acceso a usuarios sobre aplicaciones.

### 3.2.2 PERFIL GESTOR DE APLICACIONES

Este perfil solo tendrá acceso a gestionar las aplicaciones existentes en el sistema, podrá también otorgar permisos sobre aplicaciones existentes a usuarios.

### 3.2.3 PERFIL USUARIO

Únicamente tendrá acceso a las aplicaciones visibles para este usuario.

## 3.3 Interfaz de Usuario

La interfaz será simple y funcional, ya que solo se utilizará a nivel interno en cada empresa. Visualizada en un navegador web, con un menú principal en el que se tendrá acceso a las diferentes funcionalidades, estando ocultas las que no pertenezcan al perfil del usuario.

## 3.4 Software

Al ser una aplicación web, ésta será multiplataforma, pudiendo funcionar sobre cualquier navegador actual, ya que cumple los estándares de la W3C.

Como lenguaje de servidor la aplicación utiliza Python, se toma la decisión de utilizarlo por la amplia documentación que hay disponible, por el

conocimiento del desarrollador, además de la multitud de librerías que existen para simplificar su utilización. Además se ha utilizado el framework MVC Django, que simplifica muchas tareas que de implementarlas únicamente con Python sin la ayuda de ninguna librería se harían muy tediosas.

Para las vistas se ha utilizado HTML, CSS y JavaScript, además de Bootstrap para simplificar el diseño de la aplicación, que es algo que escapa al alcance de este proyecto.

En la parte de los datos se ha usado MySQL como SGBD, utilizando Django ORM para abstraer el uso de la base de datos dentro de la aplicación.

# Chapter 4

## Análisis

### 4.1 Metodología de desarrollo

Para la realización del proyecto y su documentación se ha utilizado el Rational Unified Process (RUP), junto con el Lenguaje Unificado de Modelado (UML). Se ha elegido este sistema ya que es la metodología estándar más utilizada, además de ser un grupo de metodologías que se adaptan muy bien a las necesidades de un producto.

### 4.2 Especificación de requisitos del sistema

A continuación se enumeran los requisitos funcionales que se consideran fundamentales para el sistema. Éstos serán detallados utilizando casos de uso, describiendo tanto su escenario principal como sus posibles flujos alternativos. Además se detallará cada caso de uso con su diagrama de secuencia correspondiente.

#### 4.2.1 GESTIÓN DE INSTALACIÓN

Una vez finalizada la instalación de la aplicación el administrador de la empresa debe terminar la configuración del sistema.

#### 4.2.1.1 Caso de uso: Finalizar instalación

- **Descripción:** Caso de uso para la primera vez que se accede al sistema.
- **Actores:** Administrador de la empresa.
- **Precondiciones:** La aplicación ha sido instalada.
- **Postcondiciones:** La aplicación queda configurada para su uso.
- **Escenario principal:**
  - El sistema muestra un formulario al usuario para que introduzca los datos.
  - El administrador introduce el nombre de la empresa, su email y su contraseña, que será la contraseña del administrador del sistema.
  - El sistema valida los datos y muestra al usuario un formulario para introducir usuarios junto con su rol de acceso.
  - El administrador repite el paso anterior hasta que termine de introducir usuarios
  - El sistema manda un mail a todos los usuarios introducidos para terminar su configuración
  - El sistema queda configurado.
- **Escenarios alternativos:**
  - Alguno de los datos introducidos es inválido.
    - \* El sistema indica el error y el caso de uso vuelve al paso anterior.
  - El administrador decide dejar el proceso de introducción de usuarios para más tarde.
    - \* El caso de uso termina.
  - El administrador introduce un email que ya ha introducido previamente.



Figure 4.1: Diagrama de casos de uso de Gestión de instalación

- \* El sistema indica el error y el caso de uso vuelve al paso anterior.

## 4.2.2 GESTIÓN DE USUARIOS

### 4.2.2.1 Caso de uso: Añadir usuario

- **Descripción:** Caso de uso para la creación de un usuario.
- **Actores:** Administrador de empresa.
- **Precondiciones:** El administrador se ha identificado correctamente en el sistema.
- **Postcondiciones:** Se crea un usuario con el perfil correspondiente.
- **Escenario principal:**
  - El administrador selecciona una empresa existente para añadir un usuario en ella.
  - El administrador introduce los datos del usuario y el nivel de privilegios.
  - El sistema valida que los datos son correctos y no hay ningún usuario con el mismo email.
  - El sistema crea el usuario y envía un mail al usuario para terminar la configuración
- **Escenarios alternativos:**
  - Alguno de los datos no es correcto.
    - \* El sistema indica el error y el caso de uso vuelve al paso anterior.
  - Ya existe algún usuario con el mismo email.
    - \* El sistema indica el error y el caso de uso vuelve al paso anterior.
  - En cualquier momento el administrador decide cancelar el proceso.
    - \* El caso de uso finaliza.

#### 4.2.2.2 Caso de uso: Terminar configuración de usuario

- **Descripción:** El usuario registrado termina su configuración.
- **Actores:** Usuario
- **Precondiciones:** El usuario ha sido creado previamente por un administrador y el usuario ha recibido un email con un enlace.
- **Postcondiciones:** El usuario queda configurado y con acceso al sistema.
- **Escenario principal:**
  - El usuario abre el link que ha recibido por email.
  - El sistema muestra un formulario para configurar la contraseña y el resto de datos necesarios.
  - El usuario introduce los datos.
  - El sistema valida los datos.
  - El sistema guarda los datos y da acceso al usuario.
- **Escenarios alternativos:**
  - Alguno de los datos es incorrecto
    - \* El sistema lo indica y vuelve al paso anterior.

#### 4.2.2.3 Caso de uso: Editar usuario

- **Descripción:** Caso de uso para la edición de un usuario.
- **Actores:** Usuario.
- **Precondiciones:** El usuario que se intenta editar coincide con el identificado en el sistema o bien el usuario identificado es un administrador.
- **Postcondiciones:** Se actualizan los datos del usuario.
- **Escenario principal:**
  - El sistema muestra los datos actuales del usuario.
  - El usuario modifica sus datos.
  - El sistema valida que los datos introducidos son correctos y no hay ningún otro usuario con el mismo email.



- El usuario elige guardar los datos.
- El sistema modifica el usuario.
- **Escenarios alternativos:**
  - Alguno de los datos no es correcto.
    - \* El sistema indica el error y el caso de uso vuelve al paso anterior.
  - Ya existe algún usuario con el mismo email.
    - \* El sistema indica el error y el caso de uso vuelve al paso anterior.
  - En cualquier momento el administrador decide cancelar el proceso.
    - \* El caso de uso finaliza.

### 4.2.3 GESTIÓN DE EMPRESAS

A continuación se especifican los casos de uso necesarios para llevar a cabo la gestión de las empresas clientes del sistema.

*Diagrama de casos de uso de Gestión de empresas*

#### 4.2.3.1 Caso de uso: Añadir empresa

- **Descripción:** Caso de uso para añadir una empresa
- **Actores:** Administrador del sistema.
- **Precondiciones:** El usuario tiene nivel de administrador del sistema.
- **Postcondiciones:** La empresa queda registrada en el sistema
- **Escenario principal:**
  - El administrador introduce los datos de la empresa.
  - El sistema valida que los datos son correctos y no hay ninguna empresa con el mismo nombre.
  - El administrador selecciona los productos a los que tendrá acceso la empresa.

- El sistema crea la empresa.
- **Escenarios alternativos:**
  - Alguno de los datos no es correcto.
    - \* El sistema indica el error y el caso de uso vuelve al paso anterior.
  - Ya existe alguna empresa con el mismo nombre.
    - \* El sistema indica el error y el caso de uso vuelve al paso anterior.
  - En cualquier momento el administrador decide cancelar el proceso.
    - \* El caso de uso finaliza.

#### 4.2.3.2 Caso de uso: Editar empresa

- **Descripción:** Caso de uso para la edición de una empresa.
- **Actores:** Administrador del sistema.
- **Precondiciones:** El usuario tiene privilegios de administrador del sistema.
- **Postcondiciones:** Se actualizan los datos de la empresa
- **Escenario principal:**
  - El sistema muestra los datos actuales de la empresa
  - El usuario modifica sus datos.
  - El sistema valida que los datos introducidos son correctos y no hay ningún otro usuario con el mismo email.
  - El usuario modifica los productos a los que tiene acceso la empresa.
  - El usuario elige guardar los datos.
  - El sistema modifica la empresa.
- **Escenarios alternativos:**
  - Alguno de los datos no es correcto.
    - \* El sistema indica el error y el caso de uso vuelve al paso anterior.

- En cualquier momento el administrador decide cancelar el proceso.

\* El caso de uso finaliza.

#### 4.2.3.3 Caso de uso: Ver empresa.

- **Descripción:** Caso de uso para la edición de una empresa.
- **Actores:** Usuario del sistema.
- **Precondiciones:** El usuario tiene privilegios de usuario del sistema.
- **Postcondiciones:** Se muestran los datos de la empresa
- **Escenario principal:**
  - El usuario elige una empresa de las que tiene acceso.
  - El sistema muestra los datos actuales de la empresa.
  - El sistema muestra los usuarios de la empresa.
  - El sistema muestra los productos contratados por la empresa.

#### 4.2.4 GESTIÓN DE PRODUCTOS

##### 4.2.4.1 Caso de uso: Seleccionar producto

- **Descripción:** Caso de uso abstracto incluido en otros casos de uso para seleccionar una aplicación de una lista de disponibles
- **Actores:** Gestor de aplicaciones del sistema
- **Precondiciones:** El usuario identificado en el sistema es un gestor de aplicaciones.
- **Postcondiciones:** Se selecciona una aplicación para su uso en otra finalidad.
- **Escenario principal:**
  - El sistema muestra un listado de las aplicaciones disponibles.
  - El usuario selecciona la aplicación deseada.
- **Escenarios alternativos:**
  - No hay ninguna aplicación registrada.

\* El sistema indica el error y el caso de uso finaliza.

#### 4.2.4.2 Caso de uso: Registrar aplicación

- **Descripción:** Registra una nueva aplicación en el sistema.
- **Actores:** Gestor de aplicaciones del sistema
- **Precondiciones:** El usuario identificado en el sistema es un gestor de aplicaciones.
- **Postcondiciones:** La aplicación queda registrada.
- **Escenario principal:**
  - El gestor introduce el código, el nombre y todos los demás datos de la aplicación.
  - El sistema comprueba que los datos cumplen el formato.
  - El sistema confirma el alta de la aplicación mostrando un mensaje.
- **Escenarios alternativos:**
  - Alguno de los datos introducidos tiene un formato incorrecto.
    - \* El sistema indica el error y se vuelve al paso anterior.
  - Falta algún campo obligatorio.
    - \* El sistema indica el error y se vuelve al paso anterior.
  - Ya existe alguna aplicación con ese código o nombre,
    - \* El sistema indica el error y se vuelve al paso anterior.
  - El gestor decide cancelar el registro en cualquier momento
    - \* El caso de uso finaliza

#### 4.2.4.3 Caso de uso: Editar aplicación

- **Descripción:** Edita una aplicación existente en el sistema modificando sus datos
- **Actores:** Gestor de aplicaciones del sistema
- **Precondiciones:** El usuario identificado en el sistema es un gestor de aplicaciones.

- **Postcondiciones:** La aplicación queda modificada.
- **Escenario principal:**
  - Se realiza el caso de uso *seleccionar aplicación*
  - El sistema muestra sus datos actuales, permitiendo su edición.
  - El usuario modifica los datos.
  - El sistema comprueba que los datos son correctos.
  - El sistema confirma la modificación de la aplicación mostrando un mensaje.
- **Escenarios alternativos:**
  - Alguno de los datos introducidos tiene un formato incorrecto.
    - \* El sistema indica el error y se vuelve al paso anterior.
  - Falta algún campo obligatorio.
    - \* El sistema indica el error y se vuelve al paso anterior.
  - Ya existe alguna aplicación con ese código o nombre,
    - \* El sistema indica el error y se vuelve al paso anterior.
  - El gestor decide cancelar el registro en cualquier momento
    - \* El caso de uso finaliza

#### 4.2.4.4 Caso de uso: Borrar aplicación

- **Descripción:** Borra una aplicación del sistema.
- **Actores:** Gestor de aplicaciones del sistema
- **Precondiciones:** El usuario identificado en el sistema es un gestor de aplicaciones.
- **Postcondiciones:** La aplicación queda eliminada.
- **Escenario principal:**
  - Se realiza el caso de uso *seleccionar aplicación*
  - El sistema muestra un diálogo de confirmación.
  - El usuario confirma que quiere eliminar la aplicación.
  - El sistema elimina la aplicación.
  - El sistema confirma la eliminación de la aplicación mostrando un mensaje.

- **Escenarios alternativos:**
  - El usuario selecciona que no desea eliminar la aplicación
    - \* El caso de uso se reinicia.
  - El gestor decide cancelar la eliminación en cualquier momento
    - \* El caso de uso finaliza

#### 4.2.4.5 Caso de uso: Ver detalle de aplicación

- **Descripción:** Muestra los datos de una aplicación en detalle, así como sus credenciales.
- **Actores:** Gestor de aplicaciones del sistema
- **Precondiciones:** El usuario identificado en el sistema es un gestor de aplicaciones.
- **Postcondiciones:** Los datos de la aplicación se muestran por pantalla
- **Escenario principal:**
  - Se realiza el caso de uso *seleccionar aplicación*
  - El sistema muestra los datos de la aplicación y sus credenciales.
- **Escenarios alternativos:**
  - El gestor decide cancelar el proceso en cualquier momento
    - \* El caso de uso finaliza

#### 4.2.4.6 Caso de uso: Editar usuarios de la empresa con acceso a la aplicación

- **Descripción:** Cambia los usuarios de una empresa que tienen acceso a la aplicación.
- **Actores:** Administrador de empresa.
- **Precondiciones:** El usuario identificado en el sistema es un administrador de una empresa.
- **Postcondiciones:** Los usuarios con acceso a la aplicación quedan modificados.

- **Escenario principal:**
  - Se realiza el caso de uso *seleccionar aplicación*
  - El usuario selecciona el rol que quiere editar.
  - El usuario selecciona los usuarios de su empresa a añadir a la aplicación.
  - El usuario vuelve al paso 2 para seleccionar otro rol hasta que haya acabado con todos los roles.
  - El usuario selecciona guardar.
  - El sistema modifica los datos
- **Escenarios alternativos:**
  - El gestor decide cancelar el proceso en cualquier momento
    - \* El caso de uso finaliza

## 4.3 Integraciones

### 4.3.0.1 Caso de uso: Registrar aplicación externa

- **Descripción:** Registra una nueva aplicación externa en el sistema.
- **Actores:** Administrador de aplicaciones.
- **Precondiciones:** El usuario identificado en el sistema es un administrador de aplicaciones.
- **Postcondiciones:** La aplicación externa queda registrada.
- **Escenario principal:**
  - El usuario introduce los datos de la aplicación.
  - El sistema comprueba que son correctos.
  - El usuario introduce la url externa para hacer la integración.
  - El sistema prueba la conexión.
  - El sistema guarda los datos.
- **Escenarios alternativos:**
  - Los datos son incorrectos.
    - \* El sistema lo indica y vuelve al paso anterior.

- La conexión con la url externa no se puede realizar.
  - \* El sistema lo indica y vuelve al paso anterior.
- El gestor decide cancelar el proceso en cualquier momento
  - \* El caso de uso finaliza

#### 4.3.0.2 Caso de uso: Obtener token de usuario desde aplicación interna

- **Descripción:** Un usuario obtiene un token a través de una aplicación interna.
- **Actores:** Usuario
- **Precondiciones:** El usuario existe en el sistema y tiene acceso a la aplicación.
- **Postcondiciones:** La aplicación recibe el token de usuario.
- **Escenario principal:**
  - La aplicación redirige al usuario a la web del sistema para identificarse.
  - El usuario se identifica en el sistema.
  - El sistema comprueba los datos son correctos.
  - El sistema comprueba que el usuario tiene acceso a la aplicación.
  - El sistema devuelve el token a la aplicación.
- **Escenarios alternativos:**
  - Las credenciales son incorrectas.
    - \* El sistema lo indica y vuelve al paso anterior.
  - El usuario identificado no tiene acceso a la aplicación.
    - \* El sistema lo indica y el caso de uso termina
  - El usuario decide cancelar el proceso en cualquier momento
    - \* El caso de uso finaliza



#### 4.3.0.3 Caso de uso: Obtener token de usuario desde aplicación externa

- **Descripción:** Un usuario obtiene un token a través de una aplicación externa.
- **Actores:** Usuario
- **Precondiciones:** El usuario existe en el sistema y tiene acceso a la aplicación.
- **Postcondiciones:** La aplicación recibe el token de usuario.
- **Escenario principal:**
  - La aplicación redirige al usuario a la web del sistema para identificarse.
  - El usuario se identifica en la aplicación.
  - El sistema redirige a la web de la aplicación externa.
  - El usuario se identifica en la aplicación externa.
  - El sistema devuelve el token a la aplicación.
- **Escenarios alternativos:**
  - Las credenciales son incorrectas.
    - \* El sistema lo indica y vuelve al paso anterior.
  - El usuario identificado no tiene acceso a la aplicación.
    - \* El sistema lo indica y el caso de uso termina
  - El usuario decide cancelar el proceso en cualquier momento
    - \* El caso de uso finaliza

#### 4.3.0.4 Caso de uso: Importar usuarios desde módulo externo

- **Descripción:** Se importan usuarios desde módulo externo
- **Actores:** Administrador de empresa
- **Precondiciones:** El usuario existe en el sistema y es administrador de una empresa
- **Postcondiciones:** Los usuarios quedan cargados en la aplicación

- **Escenario principal:**
  - El usuario selecciona el método de importación.
  - El usuario carga los datos siguiendo el método adecuado.
  - El sistema registra los usuarios en el sistema.

#### 4.3.0.5 Caso de uso: Sincronizar usuarios con módulo externo

- **Descripción:** Se sincroniza con una api externa para cargar los usuarios.
- **Actores:** Administrador de empresa
- **Precondiciones:** El usuario existe en el sistema y es administrador de una empresa
- **Postcondiciones:** Los usuarios quedan cargados en la aplicación
- **Escenario principal:**
  - El usuario selecciona el método de importación.
  - El usuario introduce las apis necesarias con las credenciales.
  - El sistema sincroniza con la api externa.

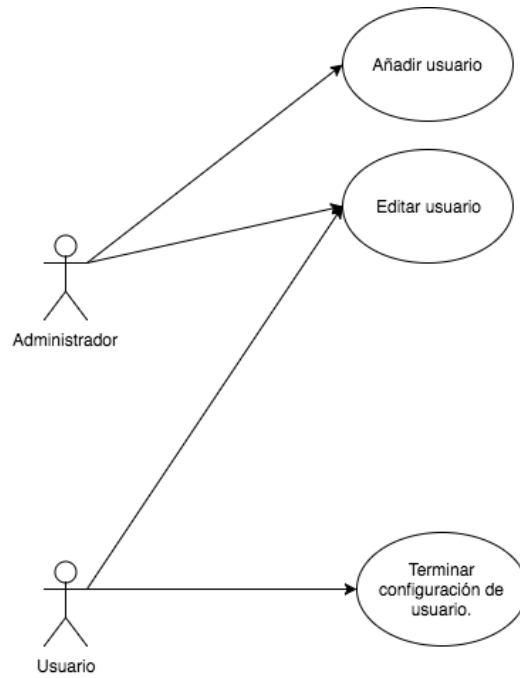


Figure 4.2: Diagrama de casos de uso de Gestión de usuarios

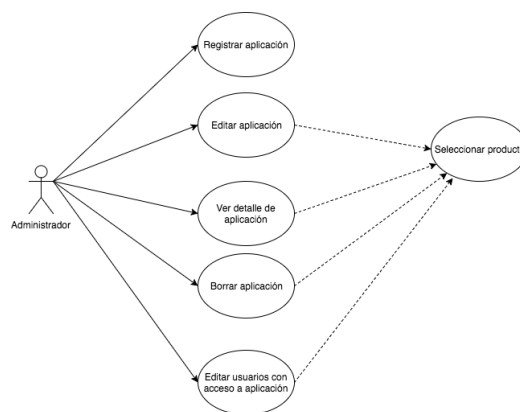


Figure 4.3: Diagrama de casos de uso de Gestión de productos

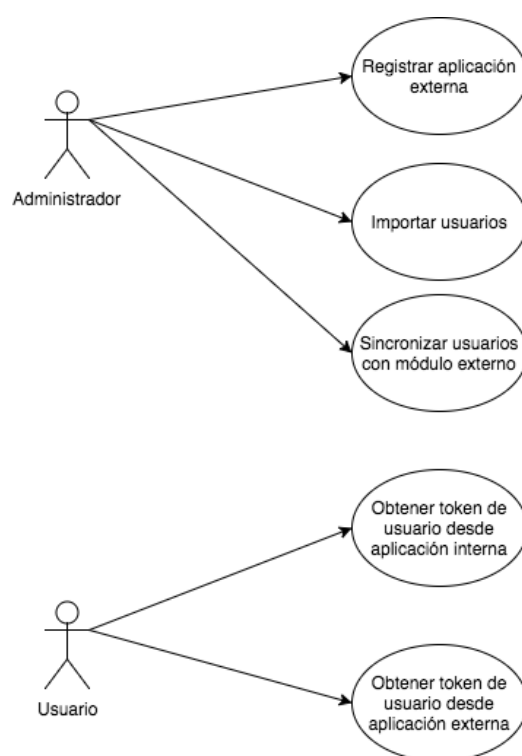


Figure 4.4: Diagrama de casos de uso de integraciones

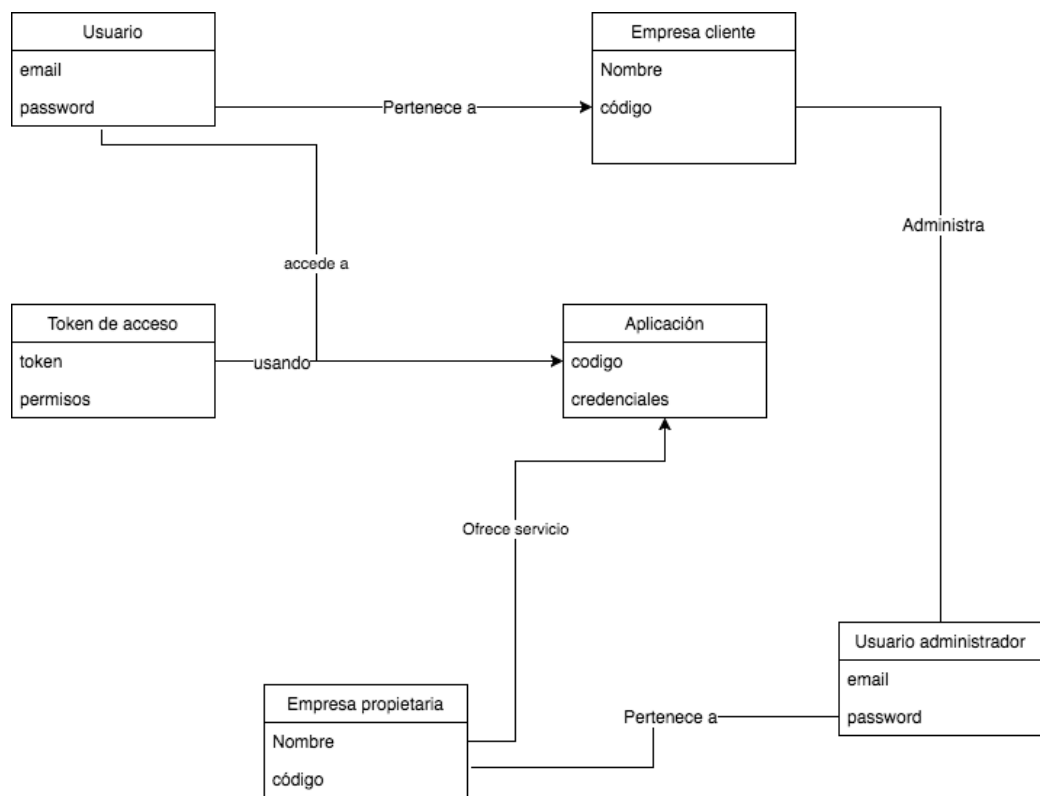


Figure 4.5: Diagrama de clases conceptual

# Chapter 5

## Diseño

### 5.1 Introducción

La fase de diseño consiste en aplicar una serie de técnicas para transformar los requisitos elicitados en la fase de análisis en una estructura detallada para el sistema de forma que se pueda implementar fácilmente a partir de ese diseño.

Este diseño ha de realizarse a varios niveles de detalle, por un lado a nivel de arquitectura interna de la aplicación. Por otro lado el diseño del modelo de datos y finalmente a más alto nivel detallar todos los componentes del sistema y como interactúan entre si, componiendo un sistema escalable.

Siguiendo los requisitos de la fase anterior, esta tarea es relativamente sencilla y debe resultar en una serie de diagramas y especificaciones que sirvan como guión y documentación a las personas que vayan a participar, ahora o en un futuro en el desarrollo del sistema.

Este documento debe contener también un modelo de despliegue incluyendo infraestructura, aplicación y configuración.

## 5.2 Arquitectura de aplicación

La aplicación se rige por el patrón arquitectónico MVC. Este patrón permite separar la lógica de negocio de la presentación, así como de los datos.

- El modelo representa a las estructuras de datos. Las clases del modelo contienen funciones para modificar los datos, insertar y actualizar la base de datos.
- La vista es la información presentada al usuario. Una vista normalmente es una página web que puede contener datos del modelo para mostrarlos al usuario.
- El controlador es un intermediario entre las dos capas anteriores y otros recursos que puedan ser necesarios. Se encarga de procesar las peticiones y generar la página web que será presentada al usuario.

Este patrón favorece la reutilización de código y la claridad.

### 5.2.1 ELECCIÓN DEL FRAMEWORK

Dado que en el lenguaje elegido para implementar la aplicación es Python, se ha elegido el framework Django, debido a la experiencia previa con esta tecnología. Este framework permite empezar a tener un software funcionando en muy poco tiempo, dedicando el trabajo casi exclusivamente a implementar el modelo y la lógica de negocio. Dado que el framework ya es conocido, la curva de aprendizaje es mínima.

### 5.2.2 ENDPOINTS

## 5.3 Base de datos

Para el diseño de la base de datos en la que se guardarán los datos manejados por la aplicación se usará un modelo relacional. Se usará MySQL como sistema de gestión de base de datos.

### 5.3.1 MODELO ENTIDAD-RELACIÓN

### 5.3.2 TABLAS Y ATRIBUTOS

## 5.4 Arquitectura del sistema

El sistema completo se compone de varias partes, si bien la aplicación es agnóstica en cuanto a los componentes de los que depende, en este documento se detallará la tecnología elegida para cada una de ellas.

La aplicación principal con la lógica de negocio está desarrollada en Python, y depende de una base de datos, en este caso se ha elegido MySQL. Esta aplicación se descompone a su vez en una API que será la que esté expuesta públicamente y un worker para tareas que se ejecutan en segundo plano. Ambos servicios necesitan de la base de datos por lo que tienen acceso a ésta.

Las peticiones de la API más usadas deberán estar en una caché para evitar la sobrecarga de la aplicación y la Base de Datos.

Todos estos servicios estarán controlados por un servidor de aplicaciones, que expondrá un punto de entrada al cual dará acceso un servidor web.

### 5.4.1 API

La api es un servicio desarrollado en Django, este servicio expone diferentes endpoints REST para realizar diversas labores en la aplicación, dentro de este servicio reside la lógica de negocio, que no está separada de la API en si misma. En caso de que otros servicios necesitaran de esta lógica de negocio la separación sería sencilla ya que el proyecto está dividido en aplicaciones y cada una de estas aplicaciones puede extraerse a una librería separada.



### 5.4.2 BASE DE DATOS

Como sistema de gestión de bases de datos se ha elegido MySQL debido a la experiencia previa con este sistema. La base de datos es accesible con un usuario con los permisos que exclusivamente necesita la aplicación.

### 5.4.3 WORKER

Un worker es un servicio que trabaja ejecutando tareas asíncronas que va recibiendo en una cola, este worker descarga de trabajo a la API, de forma que esta pueda responder rápidamente a las peticiones que recibe, derivando los trabajos más pesados a esta cola. Este worker puede recibir tareas que ejecutará inmediatamente, o bien recibir tareas pospuestas para un momento concreto en el tiempo. Este sistema debe poder funcionar de forma distribuida, por lo tanto para funcionar necesita de un backend común para almacenar las tareas hasta que estas sean ejecutadas.

Como tecnología para el worker se ha elegido celery, ya que es la mejor opción existente actualmente en Python, permite funcionar con todos los requisitos expuestos anteriormente.

Como tecnología para el backend se ha elegido redis, que es un almacén de datos clave valor que se ejecuta de forma ligera.

### 5.4.4 SERVIDOR DE APLICACIONES

Un servidor de aplicaciones es una pieza de software necesaria para ejecutar cierto tipo de aplicaciones y servir las a un cliente a través de un puerto. Django y en general Python utilizan el protocolo Wsgi para comunicarse con servidores de este tipo

WSGI es una especificación que actúa de interfaz entre servidores web y aplicaciones web, es el estándar adoptado por Python para este tipo de comunicaciones.

Entre la aplicación y el servidor puede existir un middleware para procesar

la petición y enrutarla a la parte de código correspondiente. En nuestro modelo este middleware sería el router de Django, que se encarga de parsear la URL y enviarla a la vista correspondiente que se encargará de ejecutar la petición, una vez procesada, el middleware devuelve la respuesta al servidor de aplicaciones que se encarga de transmitirla al cliente.

Como implementación de servidor de aplicaciones hemos elegido uWSGI, que es una implementación del protocolo altamente configurable, que además proporciona mejor rendimiento que la mayoría de rivales.

#### 5.4.5 SERVIDOR WEB

En ocasiones es necesario un servidor web en modo reverse proxy que se encargue de enrutar la petición al servidor de aplicaciones correspondiente, como pueden existir varios servicios para responder a la petición, delante del servidor de aplicaciones se pone este reverse proxy que se encarga de enrutar la petición al servicio adecuado.

En esta ocasión hemos elegido NGINX por su facilidad de configuración. Este servidor también nos sirve para servir ficheros estáticos como imágenes o plantillas.

#### 5.4.6 INFRAESTRUCTURA

Uno de los requisitos no funcionales de la aplicación es la alta disponibilidad, por ello necesitamos un hosting confiable del cual también podamos controlar los gastos.

Además de esto, el hosting no es lo único que necesitamos, también necesitamos poder asignar direcciones DNS, un servidor de email, entre otras cosas.

En los últimos tiempos han ido ploriferando plataformas en la nube como Amazon Web Services (AWS) o Google Cloud Platform. AWS nos ofrece de serie una capa gratuita durante un año, con lo cual nos decidimos por esta plataforma, ya que ofrece todos los servicios que necesitamos.

Otro de los requisitos no funcionales es la escalabilidad, con esto nos referimos a que el servicio pueda adecuarse a los cambios en el tráfico de la aplicación. AWS ofrece de serie servicios autoescalables, de forma que la aplicación se ofrezca en servidores más potentes o se incremente el número de servidores que la sirven automáticamente. Con lo cual en este aspecto también nos puede ser útil.

#### 5.4.7 ESCALABILIDAD

Ya hemos hablado previamente de escalabilidad, pero en un software de estas características, es importante tener claros los elementos que tienen que intervenir para que el servicio pueda responder a una carga alta de tráfico.

##### 5.4.7.1 Balanceador de carga

Un balanceador de carga es un servicio que permite distribuir la carga de trabajo entre varios nodos, evitando así la sobrecarga de uno de estos nodos, lo que provocaría una caída del rendimiento del servicio. Este componente aumenta la fiabilidad del servicio a través de la redundancia, es decir el servicio está replicado en varios nodos y es el balanceador de carga el que se encarga de distribuir las peticiones a base de varios criterios, entre los que estarían la carga de cada nodo, si un nodo se identifica como no saludable, etc.

Este servicio divide la carga usando interfaces de red, por lo que actúa en la capa 4 del modelo OSI, esto quiere decir que utiliza las direcciones IP y el puerto TCP de origen y destino para enrutar la petición.

AWS ofrece Elastic Load Balancer como componente para balanceo de carga. ELB es fácilmente configurable, solo hay que añadir los hosts que servirán el tráfico y ELB se encargará de repartirlo. La pega es que ELB, al ser tan sencillo, también es bastante limitado, no ofrece autoescalabilidad, y reparte la carga por igual, por lo que si las diferentes máquinas configuradas tienen capacidad diferente, unas podrían llegar al límite de carga antes que otras. Con el nivel de carga estimado inicialmente parece suficiente con este mod-

elo, pero aun así hay que tener previsión para mover a un modelo diferente, autoescalable, que permita reducir costes. Para ello, Amazon ofrece una alternativa llamada Application Load Balancer, que es un balanceador de carga pensado para aplicaciones. Este componente actúa en la capa de aplicación, basándose en el contenido de la petición para enrutarla al objetivo correspondiente. Está pensado para arquitecturas de aplicación modernas, como microservicios y aplicaciones basadas en contenedores tipo Docker o Kubernetes, de la cual nos podríamos beneficiar para mejorar la disponibilidad.

#### **5.4.7.2 Servicios autoescalables**

Amazon ofrece el servicio Elastic Beanstalk para ejecutar aplicaciones. Este servicio es una abstracción que ofrece Amazon para desplegar aplicaciones sin preocuparnos por el provisionamiento de la máquina, autoescalado y monitorización.

#### **5.4.7.3 Alta disponibilidad**

Es importante tener en cuenta las posibles pérdidas de servicio de alguna de nuestras instancias, ni si- quiera un proveedor como AWS es 100 % fiable y hay que estar preparado para posibles fallos.

Para ello AWS ofrece diferentes availability zones donde ejecutar nuestra aplicación, de forma que podemos poner instancias de nuestra aplicación en diferentes AZ y si una de estas AZ cae, tendremos disponibles otras sin que nuestro servicio se vea afectado. En estos casos es el load balancer quién se encarga de redirigir el tráfico en caso de caídas.

### **5.5 Despliegue**

Otra parte importante del diseño de la aplicación es definir claramente como será el proceso de despliegue, esto es, como se llevará el software al entorno

de producción, como se cargará la configuración y como se preparará la infraestructura para que todo funcione.

### 5.5.1 DESPLIEGUE DE SOFTWARE

El paquete de software que se entregue tiene que ser en todo momento replicable en cualquier otro entorno, de forma que si queremos volver atrás tengamos la seguridad de que todo va a funcionar. Por tanto todas las tareas de compilación y preparación del software deben hacerse antes del despliegue. Lo que se entregará será un paquete listo para ser ejecutado en cualquier entorno.

#### 5.5.1.1 Paquetizado

Existen diferentes alternativas para este método, en nuestro caso usaremos paquetes Debian, ya que el entorno de despliegue será una máquina con esta distribución. Para preparar un paquete debian primero necesitaremos preinstalar nuestra aplicación en un entorno virtual de Python (virtualenv).

El problema principal de virtualenv es que mantiene las rutas *shebang* de los ficheros que genera de forma absoluta, por tanto mantendrá los de la máquina en la que lo instalemos, pudiendo ser diferentes a las de la máquina en producción. Para solucionar esto existe una herramienta llamada *dh-virtualenv*.

Una vez preparado el virtualenv con la aplicación instalada será eso lo que empaquetemos en el Debian, y será el paquete Debian lo que se entregará a los servidores de producción, de forma que la versión sea replicable.

#### 5.5.1.2 Gestión de configuración

Siguiendo las recomendaciones de buenas prácticas en diseño de software, la configuración no puede estar en el repositorio junto con el código. Hay diferentes razones por las que no hacer esto, algunas de ellas son que la

configuración es dependiente del entorno, por tanto tendremos tantas configuraciones como entornos tengamos, lo cual hace imposible mantener todas en el repositorio, además de esto, muchos valores de configuración son secretos, mantenerlos en claro en el repositorio no es una opción.

Para ello, la configuración se cargará siempre desde variables de entorno, para evitar de ninguna forma mantener ficheros con configuración. Estas variables de entorno se cargarán desde el software de gestión de configuración, que en nuestro caso será Ansible.

### 5.5.2 GESTIÓN DE INFRAESTRUCTURA

La infraestructura debe estar preparada y configurada para soportar nuestra aplicación, para ello hay que provisionarla con ciertos valores y necesitaremos una herramienta adecuada que nos permita automatizar el proceso para que también pueda ser replicado, para ello usaremos Puppet, además de Terraform para preparar el entorno de AWS.

## 5.6 Métricas y monitorización

### 5.6.1 MÉTRICAS

### 5.6.2 MONITORIZACIÓN

### 5.6.3 ALERTAS

# Chapter 6

## Final research study

### 6.1 Introduction

This is the introduction. Nunc lorem odio, laoreet eu turpis at, condimentum sagittis diam. Phasellus metus ligula, auctor ac nunc vel, molestie mattis libero. Praesent id posuere ex, vel efficitur nibh. Quisque vestibulum accumsan lacus vitae mattis.

### 6.2 Method

In tincidunt viverra dolor, ac pharetra tellus faucibus eget. Pellentesque tempor a enim nec venenatis. Morbi blandit magna imperdiet posuere auctor. Maecenas in maximus est.

#### 6.2.1 SUBSECTION 1

This is the first part of the methodology. Praesent mollis sem diam, sit amet tristique lacus vulputate quis. Vivamus rhoncus est rhoncus tellus lacinia, a interdum sem egestas. Curabitur quis urna vel quam blandit semper vitae a leo. Nam vel lectus lectus.

### 6.2.2 SUBSECTION 2

This is the second part of the methodology. Aenean vel pretium tortor. Aliquam erat volutpat. Quisque quis lobortis mi. Nulla turpis leo, ultrices nec nulla non, ullamcorper laoreet risus.

## 6.3 Results

These are the results. Curabitur vulputate nisl non ante tincidunt tempor. Aenean porta nisi quam, sed ornare urna congue sed. Curabitur in sapien justo. Quisque pulvinar ullamcorper metus, eu varius mauris pellentesque et. In hac habitasse platea dictumst. Pellentesque nec porttitor libero. Duis et magna a massa lacinia cursus.

## 6.4 Discussion

This is the discussion. Curabitur gravida nisl id gravida congue. Duis est nisi, sagittis eget accumsan ullamcorper, semper quis turpis. Mauris ultricies diam metus, sollicitudin ultricies turpis lobortis vitae. Ut egestas vehicula enim, porta molestie neque consectetur placerat. Integer iaculis sapien dolor, non porta nibh condimentum ut.

## 6.5 Conclusion

This is the conclusion to the chapter. Nulla sed condimentum lectus. Duis sed tempor erat, at cursus lacus. Nam vitae tempus arcu, id vestibulum sapien. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.



# Chapter 7

## Conclusion

### 7.1 Thesis summary

In summary, pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nunc eleifend, ex a luctus porttitor, felis ex suscipit tellus, ut sollicitudin sapien purus in libero. Nulla blandit eget urna vel tempus. Praesent fringilla dui sapien, sit amet egestas leo sollicitudin at.

### 7.2 Future work

There are several potential directions for extending this thesis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam gravida ipsum at tempor tincidunt. Aliquam ligula nisl, blandit et dui eu, eleifend tempus nibh. Nullam eleifend sapien eget ante hendrerit commodo. Pellentesque pharetra erat sit amet dapibus scelerisque.

Vestibulum suscipit tellus risus, faucibus vulputate orci lobortis eget. Nunc varius sem nisi. Nunc tempor magna sapien, euismod blandit elit pharetra sed. In dapibus magna convallis lectus sodales, a consequat sem euismod. Curabitur in interdum purus. Integer ultrices laoreet aliquet. Nulla vel dapibus urna. Nunc efficitur erat ac nisi auctor sodales.

# Appendix 1: Some extra stuff

Add appendix 1 here. Vivamus hendrerit rhoncus interdum. Sed ullamcorper et augue at porta. Suspendisse facilisis imperdiet urna, eu pellentesque purus suscipit in. Integer dignissim mattis ex aliquam blandit. Curabitur lobortis quam varius turpis ultrices egestas.

## Appendix 2: Some more extra stuff

Add appendix 2 here. Aliquam rhoncus mauris ac neque imperdiet, in mattis eros aliquam. Etiam sed massa et risus posuere rutrum vel et mauris. Integer id mauris sed arcu venenatis finibus. Etiam nec hendrerit purus, sed cursus nunc. Pellentesque ac luctus magna. Aenean non posuere enim, nec hendrerit lacus. Etiam lacinia facilisis tempor. Aenean dictum nunc id felis rhoncus aliquam.

## Chapter 8

## References