UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

GRADO DE INGENIERÍA EN INFORMÁTICA

Desarrollo de sistema basado en geolocalización dirigido a personas con Alzheimer y su entorno

Desarrollo de Software

Autor
ELENA CHAVES HERNÁNDEZ

Directores
CARLOS CANO GUTIÉRREZ
CARLOS RODRÍGUEZ DOMÍNGUEZ

Facultad de Educación, Economía y Tecnología de Ceuta

—

Granada, 7 de Julio de 2021

Dedicado a

...

Resumen

En este proyecto se procederá con el diseño y el desarrollo de un sistema basado en geolocalización dirigido a personas con Alzheimer y su entorno...

Abstract

Resumen en inglés...

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Motivación

La demencia se puede definir como el deterioro adquirido de las capacidades cognitivas que entorpece la realización satisfactoria de las actividades diarias. El principal causante de demencia, que se estima entre un 60 y un 70 por ciento, es la Enfermedad de Alzheimer (EA), una enfermedad degenerativa que se da habitualmente en pacientes mayores de 65 años. Es importante en este tipo de enfermedades considerar la poca información que se tiene sobre la enfermedad y que, hasta ahora, no tiene cura. Esto se traduce en que no sólo afecta al sujeto, sino también a sus cuidadores y familiares por el grado de dependencia y discapacidad que se puede llegar a sufrir durante la EA.[1]

En este trabajo, nos centramos en los pacientes que padecen la EA en una fase temprana y su entorno. En esta fase se observa, aparte de la pérdida de memoria reciente, episodios de ansiedad y falta de asociatividad, concentración, sueño y apetito, entre otros. Esto supone tanto para la persona que padece EA como para su entorno un aumento de responsabilidades en el cuidado, dándose una gran pérdida de autonomía en todas las partes implicadas y llegando a generar problemas graves de ansiedad y depresión.

Una plataforma de servicios orientada y creada especialmente atendiendo a las necesidades de todas las personas implicadas, puede ofrecer no sólo tranquilidad al llevar un seguimiento, monitorización o información relevante sobre la persona al cuidado, sino también una mayor autonomía. Todo esto se traduce en un aumento de la calidad de vida, en concreto, colocar un

pequeño dispositivo de localización sobre la persona que padece EA, disminuye la dependencia del sujeto. Utilizando IoT junto a elementos como una etiqueta NFC, puede ayudar a llevar a cabo tareas cotidianas, como olvidar qué comprar, que se empiezan a ver afectadas desde el inicio de la EA.

1.2. Objetivos

1.2.1. Objetivo General

El objetivo general de este proyecto es crear un sistema completo con geolocalización haciendo uso de tres dispositivos. Estos dispositivos incluyes un pequeño dispositivo hardware IoT especialmente diseñado para el proyecto a partir de componentes disponibles en el mercado, y dos terminales móviles. Con esto se pretende conseguir una mayor autonomía tanto para el paciente con Alzehimer como para sus cuidadores.

1.2.2. Objetivos específicos

Los objetivos específicos de este proyectos son:

- Hacer uso de los componentes hardware orientados al IoT disponibles en el mercado para construir un dispositivo que se adecúe a nuestras necesidades.
- Crear una aplicación especialmente diseñada para personas que sufren EA y su entorno.
- Unificar las tecnologías actuales para crear un sistema completo.
- Compartir con la comunidad la información, y los avances recogidos en este proyecto.

1.3. Estructura de la memoria

En este primer capítulo introductorio se exponen la motivación y los objetivos del proyecto, al igual que se los recursos utilizados y el desarrolo temporal.

En el Capítulo 2, se realiza un análisis documental sobre los fundamentos tanto teóricos como técnicos sobre los sistemas y aplicaciones existentes

cuyos objetivos se asemejan a los de este proyecto. Se conl¡cluirá con las carencias y las nuevas aportaciones para mejorar la calidad de estos sistemas y aportarle un valor añadido a nuestro proyecto.

En el Capítulo 3, se establecen los requisitos previos al diseño del sistema, así como los requisitos y la arquitectura general de este. También se añade una sección donde se contemplan los distintos Mockups diseñados según las necesidades de los distintos usuarios del sistema.

A partir de lo expuesto en el Capítulo 3, se especifican en el Capítulo 4 los distintos prototipos que se han ido construyendo y desarrollando de manera incremental hasta llegar al prototipo final.

En el Capítulo 5 se hará un análisis sobre las metas y objetivos cumplidos, así como una valoración sobre algunos aspectos y enfoques futuros de cara a mejorar el sistema.

El Capítulo 6 tiene el mismo contenido que el Capítulo 5 traducido al inglés.

1.4. Recursos utilizados

Recursos software

Angular: Framework modular y escalable diseñado para el desarrollo de aplicaciones web utilizando el patrón MVC (Modelo-Vista-Controlador). Su lenguaje de programación es TypeScript, una ampliación de JavaScript. Versión: 13.x

AdobeXD: Herramienta de edición gráfica orientada a la creación de interfaces tanto web como móviles. Se utiliza para la creación de Mockups. Versión: 36.2.32.5

Node.js: Entorno de ejecución de JavaScript diseñado para la creación de aplicaciones web escalables y orientado a eventos asíncronos. En este proyecto lo utilizaremos tanto como motor JavaScript para Angular, como para crear servicios REST. Versión: 14.15.4

MySQL Server: Sistema de gestión de bases de datos relacionales usando el modelo cliente-servidor. Versión: 8.0.23

MySQL Workbench: IDE para trabajar con el sistema SQL. Versión: 8.0

JetBrains Product Pack for Students: Paquete gratuito de IDEs ofrecido a estudiantes de mano de JetBrains. Haremos uso de PyCharm y WebStorm. Versiones: 2020.3

Django: Framework de alto nivel para el desarrollo web utilizando el patrón MVC (modelo-Vista-Controlador). Desarrollado en Python, lo utilizaremos para crear servicios. Versión: 3.1.7

Python: Lenguaje de programación interpretado. Versión: 3.8.6

JavaScript: Lenguaje de programación para web.

MongoDB: Sistema de gestión de bases de datos NoSQL usando el modelo cliente-servidor orientado a documentos de código abierto. Versión: 4.4.4

Ionic: Framework para el desarrollo de aplicaciones multiplataforma. Como framework de interfaz utilizaremos Angular con motor Node.js. Versión: 4.12

Postman: Herramienta para hacer tests de APIs REST. Versión: 8.1.0

Mendeley Desktop: Gestor de referencias bibliográficas multiplataforma. Versión 1.19.4

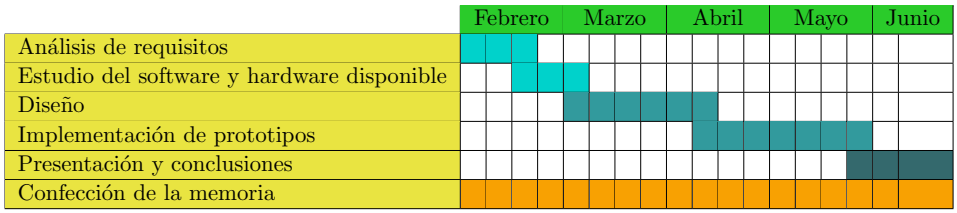
LaTeX : Sistema de composición de textos. El editor utilizado es TeXworks y la distribución MikTeX.

Visual Paradigm: Añadir descripcion.

git: Sistema de control de versiones distribuido. Los repositorios se localizan en GitHub. Versión: 2.28.0.windows.1

Recursos hardware

1.5. Planificación temporal


Febrero
Marzo
Abril
Mayo
Junio
Análisis de requisitos
Estudio del software y hardware disponible
Diseño
Implementación de prototipos
Presentación y conclusiones
Confección de la memoria

Capítulo 2

Estado del arte

2.1. Fundamentos

Glosario

2.2. Análisis de sistemas existentes

Jiobit

Jiobit es un sistema de seguimiento en tiempo real que fue diseñado por Jhon Renaldi, que tras perder a su hijo en un parque en 2015, buscó dispositivos para localizar en tiempo real, pero no encontró ningún dispositivo "wearable" adecuado. Buscaba que fuera elegante, tuviera una batería duradera y no dependiera de un dispositivo, ya fuera por necesitar de conexión bluetooth o de cualquier otro tipo.

Aunque este dispositivo fue diseñado para su uso en niños, por su versatilidad pronto se extendió a mascotas, adultos y personas mayores. Tiene el tamaño de una galleta y dispone de un botón de alerta para avisar en tiempo real, el dispositivo se conecta a un servicio en la nube que a su vez conecta con los dispositivos a los que esté asociado Jiobit.

Ventajas:

- No necesita de otro dispositivo para enviar la posición actual al servidor.
- Es ligero y fácil de colocar.
- Cuenta con un botón de aviso.
- Permite establecer sitios seguros.

Inconvenientes:

- Hay que pagar mensualmente por el servicio.
- Es fácil que el paciente se deshaga del dispositivo al ser un "wearable" bastante visible.
- Puede que se olvide la recarga del dispositivo.

Alexa Eldercare Toolbox[5]

Este sistema fue propuesto por investigadores del "Instituto Tecnológico de Stevens" en 2020, partiendo de estudios en los que la población anciana prefiere quedarse en su hogar. tomando como base el coste anual que puede suponer el cuidado de una persona mayor, se diseñó un sistema utilizando un dispositivo Amazon Echo Dot. De esta manera, también se evita el cuidado en casa, reduciendo el riesgo de contagio de COVID-19.

La potencia de este sistema es la amplia gama de necesidades que puede cubrir, algunas de las que se cubrieron con este sistema fueron: ayudar en la realización de tareas diarias a través de instrucciones, control del tiempo en la cocina, ayuda con la dieta o riesgos de caídas.

Ventajas:

- Cubre muchísimas necesidades dentro del hogar.
- Se puede configurar facilmente a través de una aplicación.
- Totalmente personalizable.
- Contacto directo en tiempo real.

Inconvenientes:

- No avisa cuando sale de casa.
- Depende de una conexión a internet.
- Puede que se olvide la recarga del dispositivo.
- Amazon Echo se debe colocar en un sitio estratégico de cada domicilio para que esté en continuo contacto.
- El paciente puede rehusarlo, puesto que es un cambio significativo para la vida cotidiana.

2.3. Análisis de aplicaciones móviles

Alzheimer Master

Esta aplicación, creada por un empresa Húngara y publicada en 2017 está disponible para Android. Su función es crear notificaciones con mensajes de voz o video personalizados y grabas la reacción de la persona afecta ante el mensaje. Su objetivo es combatir la soledad y ofrecer apoyo a la persona afectada por la enfermedad.

Existe una versión gratuita, que permite crear una notificación y una grabación con el objetivo de probar si esta se ajusta a las necesidades del usuario. En su versión de pago se eliminan las restricciones y permite conectart la aplicación con un "Android Smart Watch", añadiendo la posibilidad de realizar acciones cuando el usuario se despierta de noche y así evitar la desorientación. Como mejora de futuro, se está trabajando en añadir una función para saber cuándo sale de casa.

Ventajas:

- Notificaciones totalmente personalizadas.
- Se puede grabar cómo reacciona a los mensajes.
- Permite bloquear la aplicación con un pin para que el afectado no pueda configurarla.

Inconvenientes:

- Hay que configurar la aplicación y ver las reacciones desde el dispositivo de la persona que la va a utilizar.

- Para funcionalidades más complejas se necesita de un dispositivo Smart Watch.
- No se puede personalizar la aplicación.
- En caso de olvidar el pin no se puede acceder a la aplicación.

Timeless

Esta aplicación, creada por Emma Yang en 2018 con tan sólo 14 años, nació con el objetivo dede ayudar a su abuela. Ofrece una conexión en tiempo real entre paciente, cuidador y familia/amigos. Permite enviar fotos con etiquetas de identificación, programar eventos y, con ayuda del localizador del dispositivo, saber el tiempo, una galería de fotos de personas, la opción de identificar automáticamente a la persona a la que se le hace la fotografía desde la app y realizar llamadas a las personas que así se haya configurado.

Desde su creación, la aplicación se ha seguido desarrollando y ya se ofrece en varios idiomas para Mac, iPhone y iPad. La financiación para este proyecto que aún sigue en marcha con planes de mejora como añadir gamificación o el uso de inteligencia artificial para detectar hábitos o anomalías en el comportamiento, vino a través de la plataforma "Indiegogo", con un presupuesto inicial de 8,837.

Ventajas:

- El cuidador es el que crea y maneja los eventos del paciente.
- Las aplicaciones están sincronizadas.
- Permite estar en total contacto con el entorno en una única aplicación.
- Utiliza Inteligencia Artificial para el reconocimiento de personas dentro del entorno a partir de una fotografía.
- Se accede a la palicación a través de la huella digital o reconocimiento facial.

Inconvenientes:

- Únicamente está disponible para dispositivos Apple.
- Requiere un aprendizaje por parte del paciente para el manejo.
- No permite utilizar la ubicación para saber dónde se encuentra el paciente.

2.4. Conclusiones

Tras analizar múltiples sistemas de asistencia y aplicaciones disponibles diseñadas especialmente para nuestro público objetivo vemos que hay muy buenas ideas, pero que todas dependen de un dispositivo móvil o "*wearable*". En el caso de la aplicación "Timeless" vemos resuelto ese gran problema de la comunicación simple a través de una plicación, en la que los cuidadores manejan casi por completo toda la aplicación del paciente que sufre EA; por otro lado, Jiobit ofrece una localización en tiempo real con un dispositivo que pasa desapercibido y Alexa se puede utilizar para establecer rutinas. Cada uno de los sitemas y aplicaciones vistos están diseñados con un objetivo específico que cumplen con creces, pero ninguno de ellos hace un uso unificado de todas las tecnologías, lo que podría solventar muchas deficiencias que tienen cada uno de ellos con respecto a los otros para solucionar nuestro problema.

eHealth	Desde hace más de una década se utiliza el término eHealth consistente en, según la OMS desde 2005, *el apoyo que la utilización costoeficaz y segura de las tecnologías de la información y las comunicaciones ofrece a la salud y a los ámbitos relacionados con ella, con inclusión de los servicios de atención de salud, la vigilancia y la documentación sanitarias, así como la educación, los conocimientos y las investigaciones en materia de salud*[3]. Este término encaja perfectamente con nuestro proyecto, una plataforma de servicios ofrecido especialmente para atender la salud de los pacientes objetivos.
Internet of Things	Un término muy extendido y que engloba gran parte de los elementos que forman parte de nuestro día a día, el "IoT", del inglés, "*Internet of Things*". Hace referencia a la interconexión existente entre todo tipo de dispositivos, desde un sensor de movimiento o una cámara de vigilancia hasta cualquier tipo de dispositivo móvil complejo o incluso industrial, como puede ser un Smartphone.
Microservicios	Para realizar la conexión entre los dispositivos que forman parte de la red y la información con la que trabajan, están cada vez más extendidos los denominados "*microservicios*", que dividen el software en elementos muchos más pequeños, lo que aporta una gran escalabilidad, la posibilidad de trabjar en distintos lenguajes según el servicio que se quiera ofrecer o una mayor capacidad de recuperación, ya que los microservicios pueden trabajar de forma independiente; en contraposición, añade complejidad al diseño, ya que tienen que estar coordinados entre sí y las pruebas pueden llegar a ser algo tediosas.
Cloud Computing	 Al hablar de computación y almacenamiento de datos, podemos decir que el futuro es lo que conocemos como "Cloud Computing". Actualmente, se ofrecen todo tipo de servicios en la nube a nivel computacional; lo más importante es que únicamente se paga por los recursos utilizados y te permite de manera dinámi-

Capítulo 3

Diseño y descripción del sistema

3.1. Descripción y requisitos previos

Este sistema de información, consistente en dos aplicaciones, una dedicada a la supervisión y cuidado sobre el paciente y otra como vía de conexión entre el paciente y su entorno más cercano. El control sobre los datos recae casi en exclusiva sobre el cuidador, ya que puede ocurrir que el paciente se sienta ante una aplicación extraña y desconocida y desconfigure o cambie sin querer ciertos valores. Es por eso la importancia de un diseño completamente ubicuo adaptado en especial para nuestros usuarios objetivo.

Para el diseño visual de la aplicación dedicada en especial al paciente, he tomado como referencia a dos cuidadores de personas con alzheimer, una trabjadora en residencia de mayores y un cuidador a tiempo parcial de personas que sufren la enfermedad, así como una terapeuta ocupacional que actualmente trabaja a tiempo completo en un centro de día para mayores. Tras muchas entrevistas y experiencias vividas, la mejor solución fue una aplicación lo más sencilla posible para el usuario haciendo uso de pictogramas, ya que se pierde la capacidad de reconocer palabras antes que la de reconoces objetos. En cuanto al color elegido, todos coincidieron en un color no muy fuerte o llamativo, aunque depende bastante del apciente en concreto, por lo que se da la posibilidad a los cuidadores de cambiar tanto el color como el tamaño en el que el paciente ve laa aplicaciçon, para poder adaptarla totalmente.

La dificultad de diseñar una aplicación sencilla recayó en encontrar los elementos más sencillos, pero cuyo reconocimiento fuera practicamente automático para cualquier paciente que haga uso de ella a la vez que se mantiene un diseño gráfico suave, a la vez que cumpliese con los requisitos del sistema, entre los que se encuentra la mayor disponibilidad posible en tiempo real, al igual que interactuar con otros elementos del entorno de manera sutil, sin que supusiese un cambio de hábito; que es la base de los sistemas ubicuos.

3.1.1. Requisitos del sistema

Describimos a continuación los requisitos del sistema, divididos en dos bloques (aplicación cuidador y aplicacion paciente) y tres grupos (funcionales, no funcionales y restriccones semánticas) teniendo en cuenta nuestros principales objetivos:

1. Conectar de manera continua al cuidador con la localización del paciente.

2. Permitir al paciente contactar de forma sencilla con sus contactos más cercanos.

3. Mantener una conexión en tiempo real en tareas compartidas entre el paciente y el cuidador.

3.1.1.1. Aplicación para cuidador

Requisitos funcionales

RF1 . Iniciar sesión

El sistema debe permitir el inicio de sesión de un usuario.

- Los usuarios

RF2 . Registar usuario

El sistema debe permitir el registro de un usuario.

- El sistema permitirá crear una nueva cuenta de usuario haciendo uso de un nombre de usuario y contraseña.

- De manera no obligatoria se podrán introducir los datos pertenientes al usuario, en concreto nombre, apellidos, teléfono y dirección.
- Adicionalmente se podrá subir una imagen de perfil. En caso contrario se usará una por defecto.

RF3 . Registrar paciente

El sistema debe permitir el registro de un nuevo paciente.

- Se podrá registrar a un nuevo paciente.
- Este tipo de usuario debe poder iniciar sesión únicamente en la aplicación dedicada al paciente.

RF4 . Editar perfil

- El sistema debe permitir editar el perfil de un usuario registrado.
- El nombre de usuario no será editable.

RF5 . Editar perfil

- El sistema debe permitir editar el perfil de un usuario registrado.
- El nombre de usuario no será editable.
- Otro ejemplo para la build.

Requisitos no funcionales

Restricciones semánticas

3.1.1.2. Aplicación para paciente

3.2. Arquitectura del sistema

Debido a las características del proyecto, se eligió una arquitectura cliente-servidor, ya que hay una interacción bastante estricta y encapsulada entre los elementos que componen nuestro sistema (Figura 3.1).

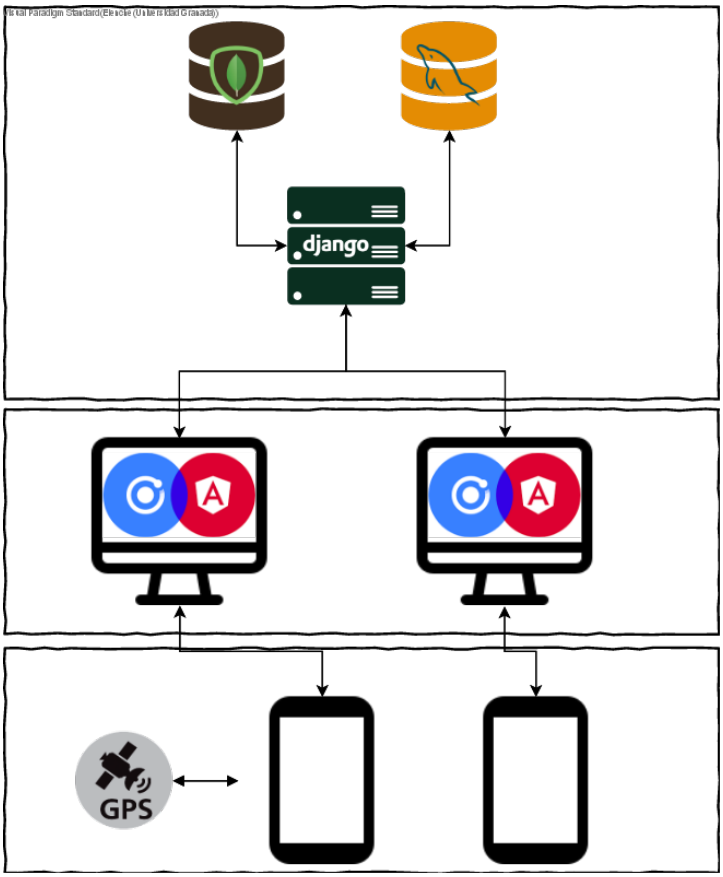


Figura 3.1: Arquitectura del software

3.2.1. Cliente

En el lado del cliente encontramos dos aplicaciones con diferentes objetivos, pero con la misma arquitectura. Hemos usado los frameworks Ionic y Angular, que juntos, nos ayudan a construir aplicaciones multiplataforma de manera sencilla. En nuestro caso, dispondremos de la aplicación para *TBD* gracias a Cordova, que junto a Ionic transforma la aplicación web

nativa; construida con Angular, en aplicaciones nativas.

Las aplicaciones Angular se basan en módulos, componentes y directivas, esto implica la reutilización de componentes y la clara diferencia entre los distintos elementos. Cada módulo contiene una plantilla HTML; donde definimos la interfaz DOM, un componente Typescript; donde se definen las clases, funciones y directivas, un módulo Typescript donde se configuran todos los elementos asociados al módulo, y un enrutador, encargador de definir las rutas o subrutas del módulo y el componente asociado. En nuestro caso hemos divido las dos aplicaciones siguiendo el mismo patrón.

Como vemos en la imagen, dividimos nuestra aplicación en: componentes, los cuales se reutilizarán en los distintos módulos; interfaces, que definen los modelos que provienen de las API; páginas, que son componentes con enrutador; servicios, donde definiremos las distintas llamadas a las API y su correspondencia de la información con los distintos modelos definidos en los servicios; y la aplicación principal, acompañada de un guard donde definimos a través de un servicio si el usuario puede acceder o no a las distintas rutas de nuestra aplicación.

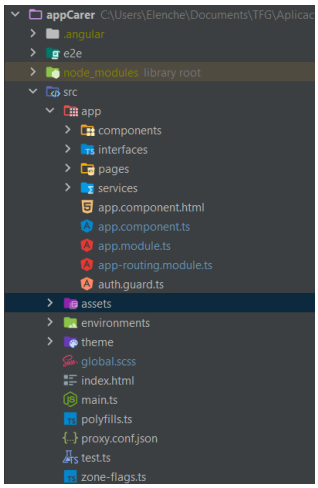


Figura 3.2: Ejemplo de arquitectura de archivos en Angular

3.2.2. Servidor

La estructura de la capa del servidor viene dada por dos elementos bien diferenciados:

3.2.2.1. Bases de datos

En nuestro servidor nos encontramos con dos tipos de bases de datos distintas, SQL y NoSQL.

Teniendo en cuenta los principios ACID, para SQL, y CAP, para NoSQL y sabiendo que SQL es una base de datos relacional, es decir, que cumple con el modelo relacional, y NoSQL es una tecnología bastante nueva, no se basa

en ningún modelo concreto y actualmente se categoriza en cinco grupos[1][4]; analizamos el rendimiento de ambos en operaciones CRUD(*create, retrieve, update, delete*)[2], llegando a la conclusión que para datos que no van a sufrir muchos cambios pero sí van a ser consultados y que necesitan ser muy consistentes, utilizaremos SQL, y para aquellos donde haya flujo continuo de datos, NoSQL.

Los sistemas gestores de bases de datos, los hemos elegido en base a su uso en el mercado, acotando únicamente a aquellos de código abierto, decidimos utilizar MySQL (SQL) para todos los datos, excepto el seguimiento de GPS, para el que utilizamos MongoDB (NoSQL) dada la necesidad de tener completa disponibilidad. Como software para gestión utilizaremos los ofrecidos de manera gratuita por cada una de ellas: MySQLWorkbench Community Edition y MongoDBCompass.

Base de datos MySQL

Teniendo en cuenta los requisitos descritos en la sección (meter sección) y el modelo relacional en el que nos basamos, podemos ver el diseño final en la Figura 3.3.

Base de datos MongoDB

Para esta base datos, el diseño se ha elegido en función de las necesidades de los cuidadores de registrar la localización de sus pacientes cada cierto período de tiempo, para ello, tenemos un documento asociado a cada paciente que contiene toda su información y que se actualizará cuando haya que añadir nueva información, definiendose el schema a seguir en los servicios a los que llamarán nuestras aplicaciones web. El *schema* que sigue cada documento es el siguiente:

```
{
 "_id": ObjectId,
 "id_patient": number,
 "register": [
  {
   "location": {
    "x": decimal,
        "y": decimal
   }
    "date": date
  }
```

[1]Clave/Valor, Documental, Tabular, Grafos, Orientada a objetos

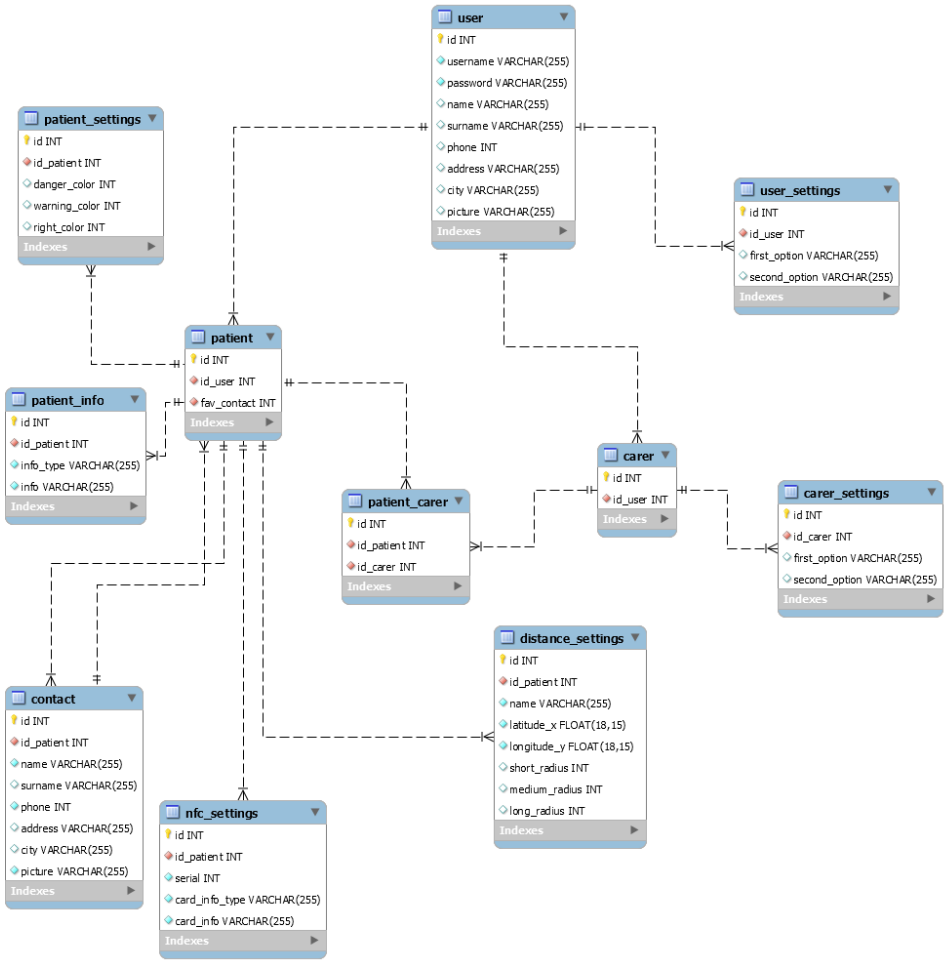


Figura 3.3: Diagrama de la base de datos SQL. Autogenerado con MySQL-Workbench.

```
 ]
}
```

3.2.2.2. Servicios

Los distintos servicios que hemos creado para la interacción entre el cliente y el servidor se dividen en dos servidores; uno dedicado a cada una de las bases de datos. La decisión de utilizar dos servidores distintos se basa más en desarrollar un aprendizaje activo como parte de la formación universitaria, ya que en cualquiera de los dos servidores se pueden ejecutar todos los servcicios aún teniendo más de una base datos. De esta forma también podemos asegurarnos, de que las peticiones se reparten y si un servidor falla, el otro seguirá activo. Todos los servicios utilizan protocolo REST, lo que facilita el desarrollo y uso de las diferentes API y simplifica la comunicación al realizarse en formato JSON.

Podemos ver las diferencias más notables entre ambos servidores en el Cuadro 3.1.

	Servidor - MySQL	Servidor - MongoDB
Lenguaje	Python	Javascript
Framework/Entorno	Django	Node.js
Framework API	Django REST Framework	ExpressJS
Arquitectura	MVT (Model View Template)	Basado en eventos

Cuadro 3.1: Cuadro comparativo entre software utilizado en los dos servidores.

Para la creación de los modelos y operaciones, se usó Django REST Framework en el caso de MySQL y la librería mongoose en el caso de MongoDB, esto es importante porque en el caso de mongoose, hay que programar las operaciones CRUD manualmente y en la de Django REST framework tenemos vistas que nos las definen automáticamente en la ruta que le hayamos asignado.

Con esta arquitectura se ve claramente que los modelos se definen en *models/* y se definen las rutas junto a las operaciones en *routers/*. El archivo *package.json* contiene la configuración de la aplicación y sus dependencias, por último, el archivo *index.js* se encarga de la conexión con la base de datos y de iniciar el servidor, proporcionando así los servicios.

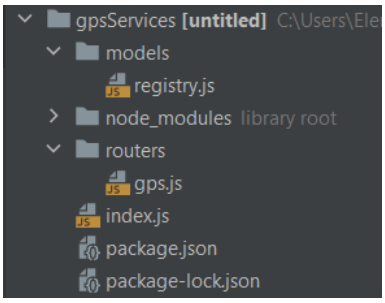


Figura 3.4: Arquitectura Servidor - MongoDB

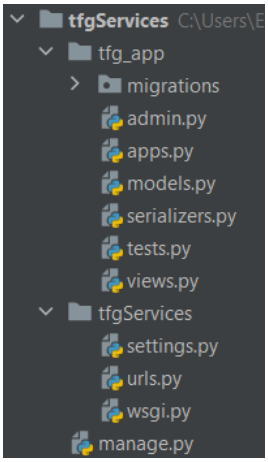


Figura 3.5: Arquitectura Servidor - MySQL.

En este caso, se sigue el modelo MVT, que conecta los *models.py* con los *serializers.py*, las *views.py* y las *urls.py*. Los serializadores definen qué campos vamos a querer de los objetos que hemos definido en los modelos; y las vistas definen, gracias al framework, qué operaciones vamos a realizar a través de las url. Los modelos se registran en la aplicación usando el archivo *admin.py*. Los archivos de configuración son *settings.py* y *wsgi.py*, y por último *manage.py* es desde donde ejecutamos el servidor.

3.3. Mockups

3.4. Conclusiones

Capítulo 4

Prototipos y desarrollo

4.1. Prototipo 1

4.2. Conclusiones

Capítulo 5

Conclusiones y mejoras futuras

5.1. Conclusiones técnicas

5.2. Conclusiones personales

5.3. Futuras mejoras

Capítulo 6

Conclusions and future works

6.1. Technical conclusions

6.2. Personal conclusions

6.3. Future works

Bibliografía

[1] Barrera López, F. J., López Beltrán, E. A., Baldivieso Hurtado, N., Maple Alvarez, I. V., López Moraila, M. A., Murillo Bonilla, L. M., Barrera-Lopez, F. J., López Beltrán, E. A., Baldivieso Hurtado, N., Maple Alvarez, I. V., López Moraila, M. A., and Murillo-Bonilla, L. M. Diagnóstico Actual de la Enfermedad de Alzheimer. *Revista de Medicina Clínica 2*, 2 (2018), 57–73.

[2] Jose, B., and Abraham, S. Performance analysis of nosql and relational databases with mongodb and mysql. *Materials today: PROCEEDINGS 24* (2020), 2036–2043.

[3] Organizacion Mundial de la Salud. 58ª Asamblea Mundial de la Salud. *58ª Asamblea Mundial de la Salud* (2005), 102–105.

[4] Palanisamy, S., and SuvithaVani, P. A survey on rdbms and nosql databases mysql vs mongodb. In *2020 International Conference on Computer Communication and Informatics (ICCCI)* (2020), pp. 1–7.

[5] Tan, K., Sekhar, K., Wong, J., Holgado, J., Ameer, M., and Vesonder, G. Alexa Eldercare Toolbox: A Smarthome Solution for the Elderly. *2020 11th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2020* (2020), 0806–0812.