Sistemas Industriales

2022 / 2023

Especificación Técnica del Supuesto, propuesto por el grupo de Enfermería.

Universitat d'Alacant
Universidad de Alicante

- Daniel Asensi Roch - 48776120C
- Elvi Mihai Sabau Sabau - 51254875L
- Jéssica Román Marroquí - 74444117V
- María Verdú Belmonte - 77039404F
- Miguel Pérez Gómez - 74393992D
- Andrea Lillo - 48788941F
- Maria Soler - 29577771R
- Helena Rocamora Torres - 29525280L
- Maria Castell Navarro - 49230349E
- Ana Maria Martinez Salinas - 29528591H

Índice

1. Descripción del proyecto

El proyecto propuesto por el grupo de medicina es el de crear un sistema capaz de predecir la carga de trabajo para un turno de un hospital, estimar de forma automática la cantidad de personal necesario para lidiar con la carga de cada turno, gestionar los turnos de los enfermeros, ser capaz de intercambiar turnos, y que se pueda interactuar de manera cómoda desde una interfaz web o móvil.

Dada esta propuesta, podemos separar el sistema en 4 módulos principales.

- Un serviddor de entrenamiento: que entrenará un modelo de machine learning usando los datos proveídos por el grupo de enfermería, capaz de estimar el personal necesario dada cierta carga.
- Un servidor de procesamiento: capaz de gestionar los turnos de cada enfermero, que se encargue de recibir los datos de la interfaz, además de mostrar los turnos actuales y una previsión de los turnos.
- Un servidor web: que muestre una interfaz de los turnos, desde la cual los enfermeros puedan ver y solicitar cambios de turno, y que envíe dichas solicitudes a un servidor de procesamiento.
- Un servidor de bases de datos: donde guardaremos los turnos asignados a para enfermero, además de la carga estimada por el modelo por cada planta y turno, y de los registros de las solicitudes de los cambios de turno.

2. Especificación Técnica

La especificación técnica de cada módulo es el siguiente:

2.1. Servidor de entrenamiento

Para realizar el entrenamiento se usarán las siguientes tecnologías:

- **Tensorflow**: Usaremos tensorflow con Keras para entrenar un modelo basado en los datos proporcionados.
- **Python**: Usaremos el lenguaje de programación Python para interactuar con tensorflow, el propio lenguaje nos da una versatilidad que otros lenguajes no nos dan a la hora de manejar herramientas de machine learning.
- **Django**: Usaremos la libreria Django para servir el modelo una vez entrenado, de esta manera nuestro modulo de procesamiento lo podrá descargar.

Además podremos continuar entrenando el modelo a medida que pasan los dias dependiendo de lo buena o mala que sea la estimación.

2.2. Servidor de Procesamiento

Para realizar el procesamiento de las peticiones del servidor web, manejar los turnos, y conectar con la base de datos, se usarán las siguientes tecnologías:

- **Node**: Node es un entorno de ejecución de código JS, usaremos Node para ejecutar código JS en el servidor.
- **ExpressJS**: Express es una librería en JS que nos permite crear desde servidores web a APIRest, esto nos permiten recibir peticiones via HTTP de otros servicios, procesarlas, y responder a dichas peticiones. Además usaremos un middleware para interceptar las peticiones, y refrescar un JWT antes de responder a las peticiones, de esta manera podremos mantener un estado de sesión al interactuar con el front web (servidor web).

- **Socket.io**: SocketIO es una librería JS que nos permite crear conexiones en vivo mediante sockets usando el protocolo TCP/IP. Esta librería la usaremos para tener además una conexión en vivo con cada cliente web (front end), de esta manera podremos ver peticiones de solicitudes, cambios en los turnos o en la carga de trabajo en vivo, sin necesidad de recargar la página.
- **SequelizeORM**: Sequelize es una librería que hace de ORM entre nuestro programa y la base de datos a usar. Esta librería nos permite interactuar de manera fácil con la base de datos, el ORM mapea las tablas de la base de datos a clases de nuestro lenguaje, haciendo fácil la interacción con la base de datos.

2.3. Servidor Web

Usaremos las siguientes tecnologías para realizar la interfaz web.

- **NGINX**: NGINX es un servidor web que hostea tanto archivos planos en html como páginas dinámicas usando SSR. Este servidor además proporciona un servicio de balanceo de cargas, y un proxy para filtrar las conexiones.
- **Svelte**: es un framework de desarrollo de aplicaciones web progresivas en JS. Usaremos este framework for su facilidad y porque proporciona una capacidad de renderizado realmente reactivo.
- **Axios**: Usaremos Axios como gestor de peticiones HTTP, con el cual interactuaremos con el servidor de procesamiento (nuestro backend). Además nos beneficiaremos de su módulo de interceptadores para JWT como medida para mantener las sesiones activas en la página web.
- **Dragable**: Es una librería creada por shopify que nos permitirá añadir la interactuabilidad del arrastre entre elementos HTML, esta librería nos servirá para modelar nuestra página web, en especifico el horario de los turnos.

2.4. Servidor de bases de datos

Usaremos en nuestro sistema 2 bases de datos.

- **PostgreSQL**: Este servidor base de datos la usaremos en conjunto con el servidor de procesamiento para guardar los turnos, sesiones, credenciales de los usuarios, plantas y otros. Hemos seleccionado esta base de datos porque nos permite la inserción masiva de datos, además de un amplio abanico de características, como (entre otros) la creación nativa de enumeradores.
- **Redis**: Este servidor base de datos la usaremos en conjunto con el servidor de entrenamiento para guardar en caché las estimaciones más recientes, y datos nuevos para entrenar actualizar y mejorar el modelo. Hemos seleccionado esta base de datos ya que nos permite guardar muchos datos de pequeño peso, además de guardarse en memoria (en RAM), haciéndola extremadamente rápida.

  Además, podemos desplegar otra base de datos REDIS para cachear los resultados del servidor de procesamiento, y así aligerar nuestra APIRest en caso de que haya una alta carga de usuarios.

3. Diseño del sistema

En este apartado se detalla el esquema del Modelo Entidad Relación que se usará en la base de datos usada por el Back End.

También se muestra el esquema de interconexión entre los módulos a desarrollar.

3.1. Esquema de las relaciones de la base de datos

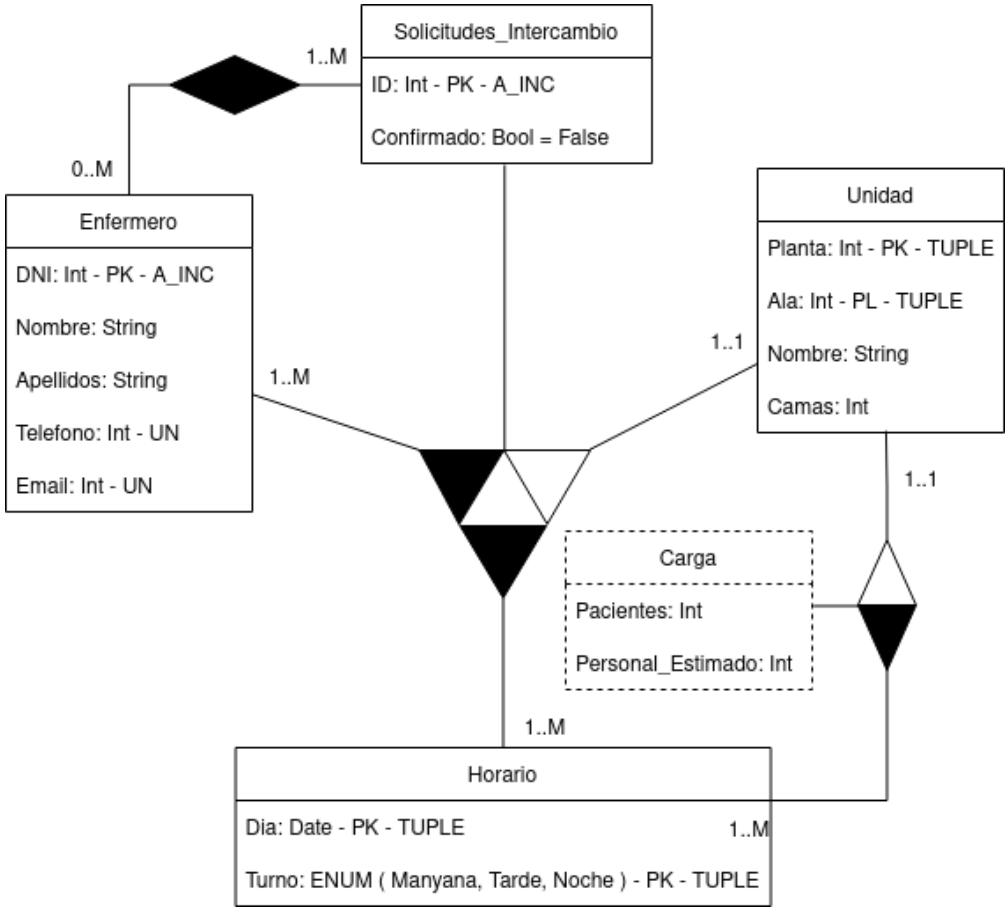

Figura 1: Diagrama EER de la base de datos (PostgreSQL).

La relación ternaria entere la unidad, el horario y el enfermero representa la relación restringida de uno o muchos enfermeros por cada planta en cada horario con su turno.

También la relación entre enfermero y Solicitudes_Intercambio representa la relación de cada enfermero al poder solicitar cambios de grupos a otros enfermeros de su misma unidad en un turno en concreto.

La carga es representada en el diagrama como un atributo de la relación entre el horario y la unidad, esta se aplica directamente al Horario, ya que dependiendo del turno en cada unidad (dependiendo de la cantidad de pacientes) se estima una cantidad de enfermeros necesarios.

3.2. Esquema de la interconexión entre los módulos

A continuación se muestra un pequeño esquema de interconexión entre los servicios descritos en los apartados anteriores.

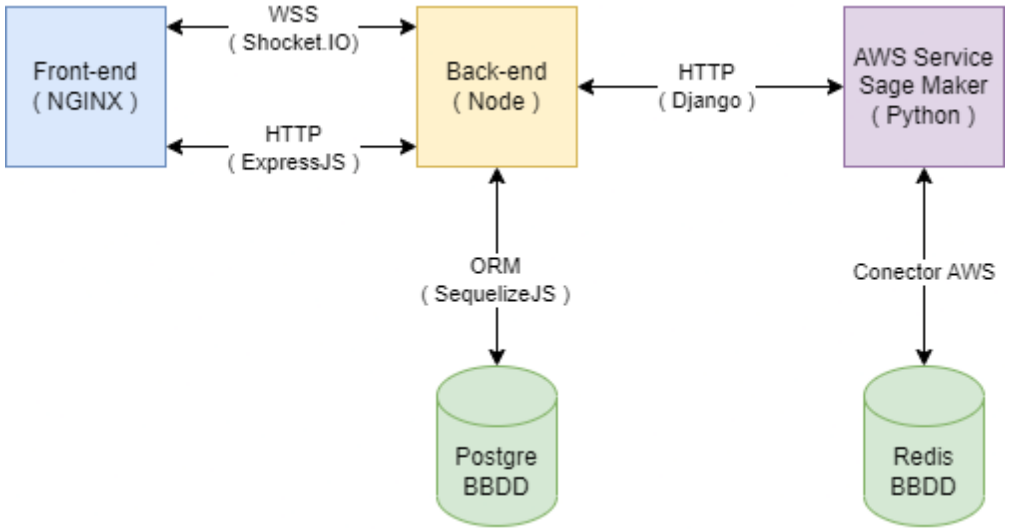

Figura 2: Esquema de interconexión entre servicios.

En este esquema de muestra la interconexión entre cada servicio y el protocolo que se usará para ello, además de la librería para dicha conexión. También especificamos el entorno / lenguaje de ejecución.

4. Desarrollo del sistema

A continuación describiremos la metodología, herramientas para el despliegue, y que servicios cloud que usaremos.

4.1. Metodología de desarrollo

La metodología a usar sera la de TDD (Test Driven Development), mediante integración continua usando Github Actions con Cicle.

4.2. Herramientas DevOps

Usaremos Docker para encapsulas cada servicio y poder desplegarlo cómodamente. También usaremos **Portainer** para tener una interfaz sencilla a la hora de manejar nuestra instalación de Docker.

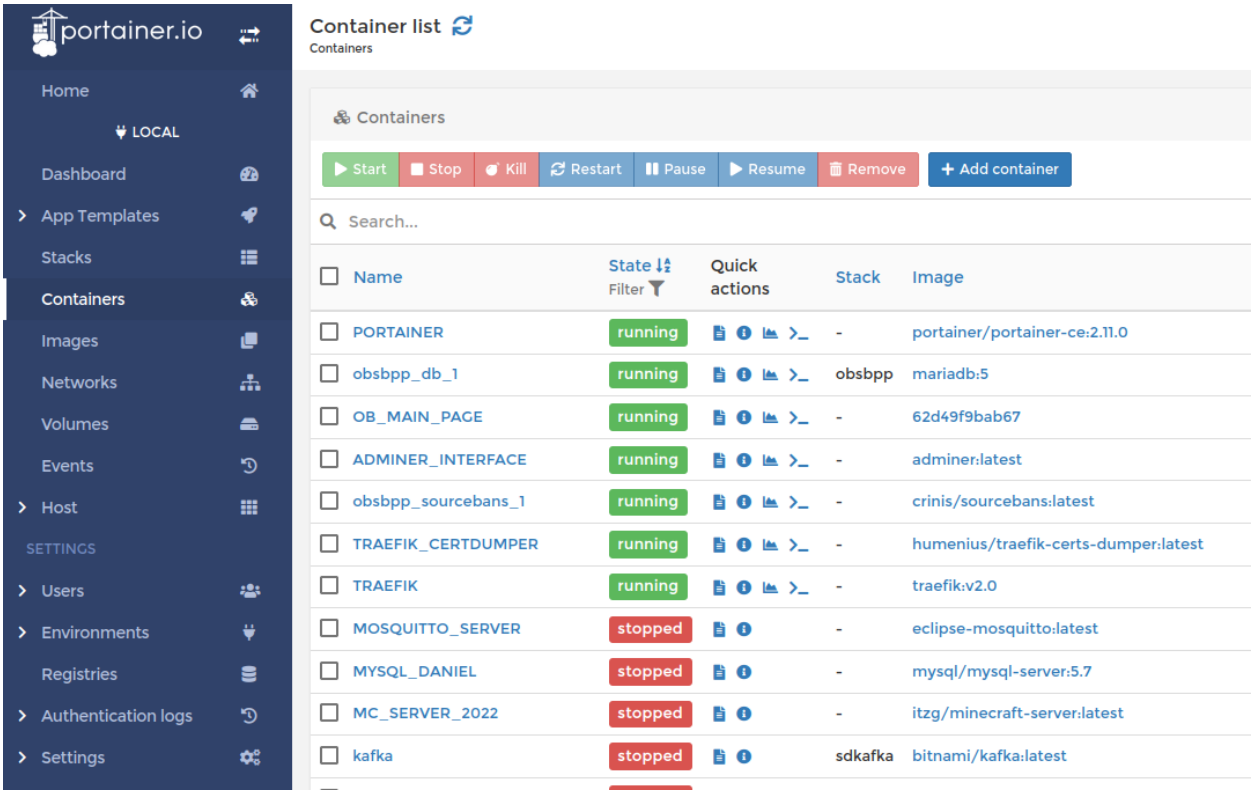

Figura 3: Interfaz Portainer con varios contenedores en ejecución.

Y para gestionar los dominios de la página web y otros usaremos Traefik Proxy, certificando los dominios y renovándolos mediante Lets Encrypt. También podemos usar Digital Ocean como entidad certificadora.

4.3. Uso de servicios en Cloud (AWS)

Usaremos un servicio de Amazon Web Services - SageMaker para realizar el entrenamiento del modelo de la estimación de la carga de trabajo.

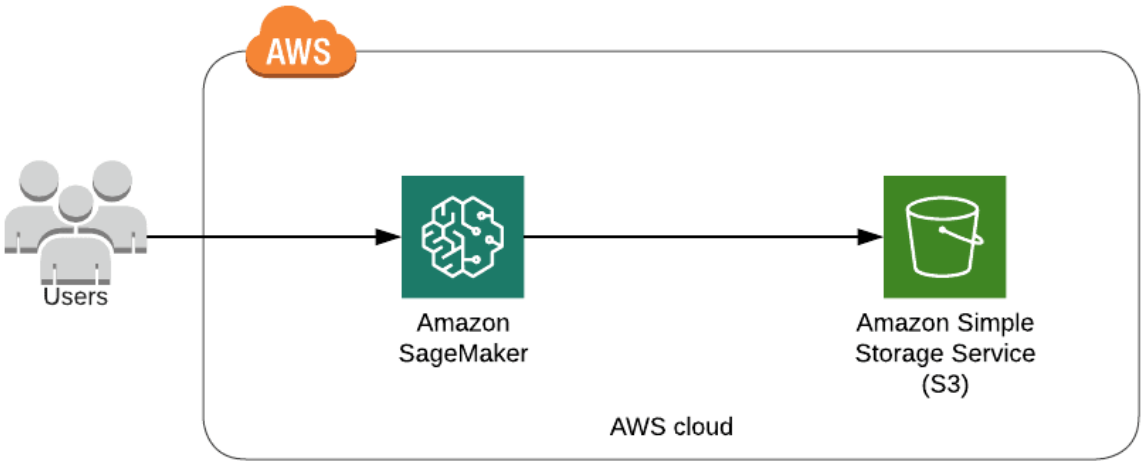

Figura 4: Esquema de interacción con AWS - SageMaker.

Y usaremos un servicios de AWS - Amazon Simple Storage Service mediante Redis para almacenar los datos más recientes en caché para el entrenamiento.