

1. Introducción

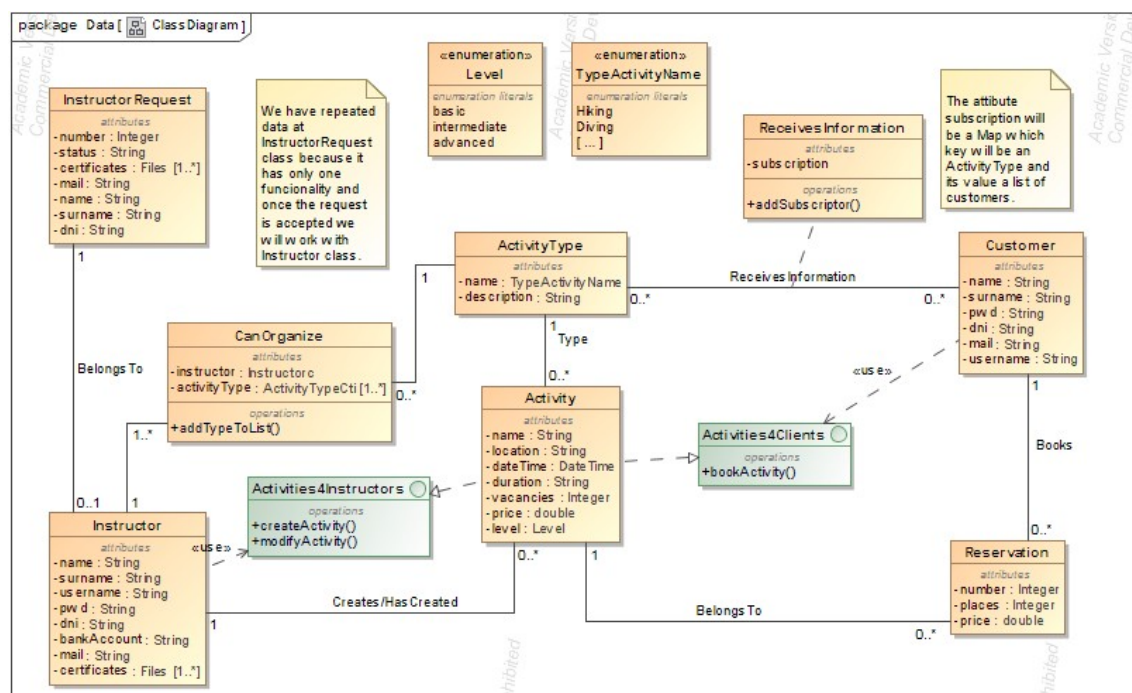
El objetivo de este proyecto es aprender a diseñar y desarrollar una aplicación con una funcionalidad lo más realista posible, conociendo y aprendiendo las técnicas usadas en un entorno profesional.

En cuanto a la organización del proyecto, esta se ha realizado a través de Trello, donde se han dividido y repartido las tareas a realizar, y se han ido indicando las tareas que iban terminando para hacer conocer al resto del equipo que dicha tarea estaba acabada. Al finalizar la realización de una tarea, esta pasaba a la fase de revisión, donde los otros dos miembros del grupo revisaban el correcto funcionamiento de dicha tarea por separado, y avisaban si había algo que no les convenciera. El objetivo de hacer esto, si bien puede parecer más lento, es asegurar que las diferentes tareas estén realizadas correctamente por parte de todos los miembros del equipo sin que una opinión pueda influir sobre otra.

Por parte de la comunicación, esta se ha realizado a través de WhatsApp, mientras que el programa se ha realizado a través del IDE Eclipse y se ha guardado en GitHub con el fin de facilitar la coordinación a la hora de seguir avanzando en su realización.

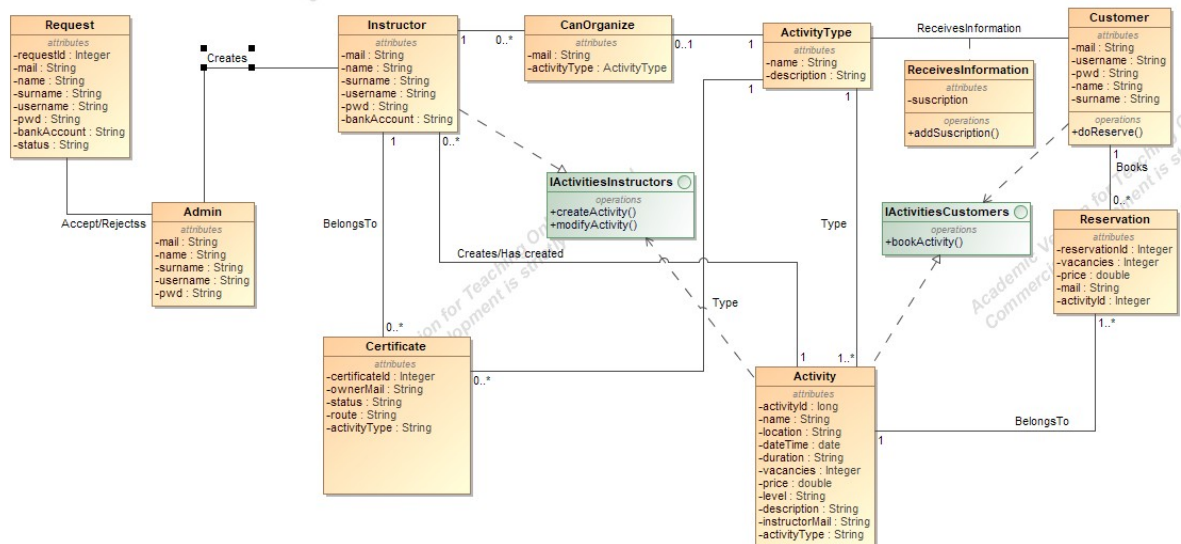
2. Diseño de la Base de Datos

El diseño de la base de datos seguido es el realizado por nosotros en la asignatura EI1023 Fundamentos de la ingeniería de software.



El motivo para elegir ese diseño y no otro ha sido que lo habíamos hecho nosotros, y por tanto, lo conocíamos.

Según hemos ido empezando a la realización del diseño de la base de datos, nos hemos ido dando cuenta de cambios que eran necesarios, como la inclusión de atributos para claves ajenas. Al final, el diseño definitivo es el siguiente:



Como se puede ver, hemos hecho una serie de cambios para simplificar el diseño, hemos eliminado las enumeraciones y hemos cambiado el tipo de los atributos que los usaban a String. Además, hemos añadido varias tablas que antes no estaban, como la del administrador o la de los certificados.

Diseño lógico:

Request(requestId, mail, name, surname, username, pwd, bankAccount, status)

Admin(mail, name, surname, pwd)

Instructor(mail, name, surname, username, pwd, bankAccount)

Certificate(certificateId, ownerMail, status, route, activityType)

Certificate.ownerMail → Instructor.maili N R P

Certificate.activityType → ActivityType.name N R P

ActivityType(name, description)

description accepts nulls

Activity(activityId, name, location, dateTime, duration, vacancies, price, level, description, instructorMail, activityType)

description accepts nulls

Activity.instructorMail → Instructor.mail N R P

Activity.activityType → ActivityType.name N R P

Reservation(reservationId, vacancies, price, mail, activityId)

Reservation.mail → Customer.mail N R P

Reservation.activityId → Activity.activityId N R P

Customer(mail, username, pwd, name, username)

CanOrganize(mail, activityName)

CanOrganize.mail → Instructor.mail N P P

CanOrganize.activityName → ActivityType.name N P P

ReceivesInformation(dni, activityType)

ReceivesInformation.mail → Customer.mail N P P

ReceivesInformation.activityType → ActivityType.activityType N P P

Diseño físico

```
CREATE TABLE Request (  
    requestId INT NOT NULL,  
        mail VARCHAR(40) NOT NULL,  
        name VARCHAR(20) NOT NULL,  
        surname VARCHAR(20) NOT NULL,  
        username VARCHAR(20) NOT NULL,  
        pwd VARCHAR(20) NOT NULL,  
        bankAccount VARCHAR(12) NOT NULL,  
        status VARCHAR(20) NOT NULL,  
        CONSTRAINT pk_requestId_request PRIMARY KEY (requestId)  
);
```

```
CREATE TABLE Admin (  
    mail VARCHAR(40) NOT NULL,  
        name VARCHAR(20) NOT NULL,  
        surname VARCHAR(20) NOT NULL,  
        username VARCHAR(20) NOT NULL,  
        pwd VARCHAR(20) NOT NULL,  
        CONSTRAINT pk_mail_admin PRIMARY KEY (mail)  
);
```

```
CREATE TABLE Instructor (  
    mail VARCHAR(40) NOT NULL,  
    name VARCHAR(20) NOT NULL,  
    surname VARCHAR(20) NOT NULL,  
    username VARCHAR(20) NOT NULL,  
    pwd VARCHAR(20) NOT NULL,  
    bankAccount VARCHAR(12) NOT NULL,  
    CONSTRAINT pk_mail_instructor PRIMARY KEY (mail)  
);
```

```
CREATE TABLE Certificate (  
    certificateId INT NOT NULL,  
    ownerMail VARCHAR(20) NOT NULL,  
    status VARCHAR(20) NOT NULL,  
    route VARCHAR(20) NOT NULL,  
    activityType VARCHAR(20) NOT NULL,  
    CONSTRAINT pk_certificateId_certificate PRIMARY KEY (certificateId),  
    CONSTRAINT fk_mail_certificate FOREIGN KEY (ownerMail) REFERENCES  
        Instructor(mail) ON UPDATE CASCADE ON DELETE RESTRICT,  
    CONSTRAINT fk_activityType_certificate FOREIGN KEY (activityType)  
        REFERENCES ActivityType(name) ON UPDATE CASCADE ON DELETE  
        RESTRICT  
);
```

```
CREATE TABLE ActivityType (  
    name VARCHAR(20) NOT NULL,  
    description VARCHAR(500),  
    CONSTRAINT pk_name_activityType PRIMARY KEY (name)  
);
```

```
CREATE TABLE Activity (  
    activityId INT NOT NULL,  
    name VARCHAR(30) NOT NULL,  
    location VARCHAR(50) NOT NULL,  
    dateTime DATE NOT NULL,  
    duration VARCHAR(20) NOT NULL,  
    vacancies INT NOT NULL,  
    price FLOAT NOT NULL,  
    level VARCHAR(20) NOT NULL,
```

```

description VARCHAR(500),
activityType VARCHAR(20) NOT NULL,
mailInstructor VARCHAR(20) NOT NULL,
    CONSTRAINT pk_activityId_activity PRIMARY KEY (activityId),
    CONSTRAINT fk_activityType_activity FOREIGN KEY (activityType) REFERENCES
ActivityType(name) ON UPDATE CASCADE ON DELETE RESTRICT,
    CONSTRAINT fk_mailInstructor_activity FOREIGN KEY (mailInstructor)
REFERENCES Instructor(mail) ON UPDATE CASCADE ON DELETE RESTRICT,
    CONSTRAINT ir_price_activity CHECK(price >= 0)
);

```

```

CREATE TABLE Customer (
    mail VARCHAR(20) NOT NULL,
    username VARCHAR(20) NOT NULL,
    pwd VARCHAR(10) NOT NULL,
    name VARCHAR(20) NOT NULL,
    surname VARCHAR(20) NOT NULL,
    CONSTRAINT pk_mail_customer PRIMARY KEY (mail)
);

```

```

CREATE TABLE Reservation (
    reservationId INT NOT NULL,
    vacancies INT NOT NULL,
    price FLOAT NOT NULL,
    mail VARCHAR(20) NOT NULL,
    activityId INT NOT NULL,
    status VARCHAR(20) NOT NULL,
    CONSTRAINT pk_reservationId_reservation PRIMARY KEY (reservationId),
    CONSTRAINT fk_mail_reservation FOREIGN KEY (mail) REFERENCES
Customer(mail) ON UPDATE CASCADE ON DELETE RESTRICT,
    CONSTRAINT fk_activityId_reservation FOREIGN KEY (activityId) REFERENCES
Activity(activityId) ON UPDATE CASCADE ON DELETE RESTRICT
);

```

```

CREATE TABLE CanOrganize (
    mail VARCHAR(20),
    activityName VARCHAR(20),

```

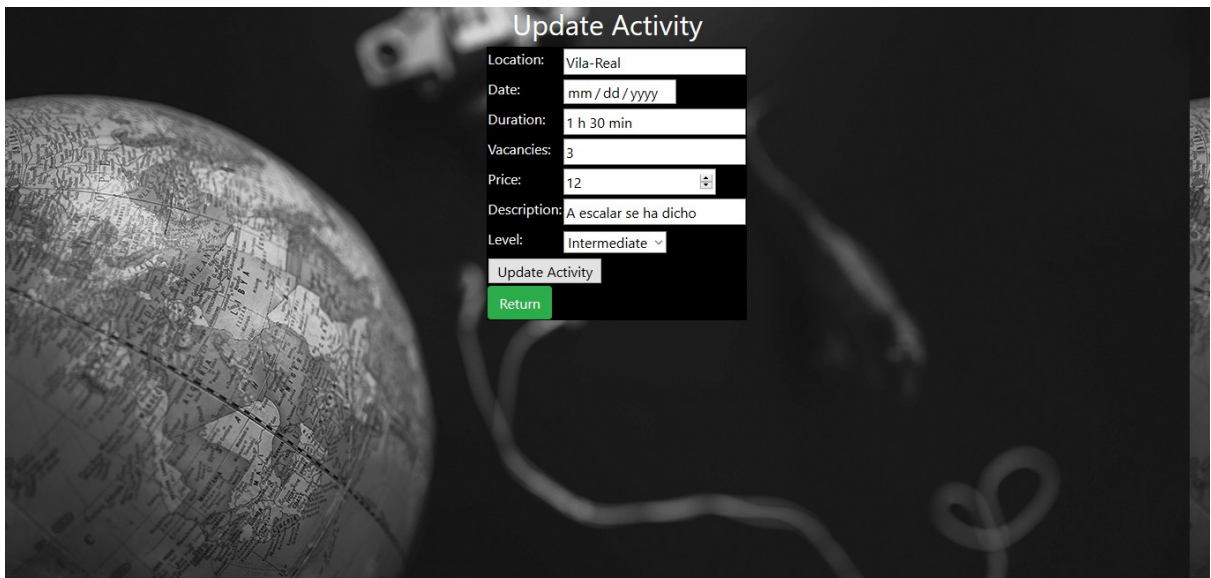
```

status VARCHAR(20),
CONSTRAINT pk_mail_canOrganize PRIMARY KEY (mail),
CONSTRAINT fk_co_mail FOREIGN KEY (mail) REFERENCES Instructor(mail) ON
UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT fk_co_activityName FOREIGN KEY (activityName) REFERENCES
ActivityType(name) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE ReceiveInformation (
mail VARCHAR(10),
activityName VARCHAR(20),
CONSTRAINT pk_mail_activityName_ri PRIMARY KEY (mail, activityName),
CONSTRAINT fk_ri_activityName FOREIGN KEY (activityName) REFERENCES
ActivityType(name) ON UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT fk_ri_mail FOREIGN KEY (mail) REFERENCES Customer(mail) ON
UPDATE CASCADE ON DELETE CASCADE
);

```

3. Diseño de Interfaces de Usuario



The screenshot shows a web application interface for updating an activity. The background features a grayscale image of a globe. The form is titled "Update Activity" and contains the following fields and controls:

- Location:** Vila-Real
- Date:** mm / dd / yyyy
- Duration:** 1 h 30 min
- Vacancies:** 3
- Price:** 12
- Description:** A escalar se ha dicho
- Level:** Intermediate (dropdown menu)
- Buttons:** "Update Activity" and "Return"

4. Implementación

Comenzamos la implementación del proyecto realizando la parte de los monitores, ya que consideramos que sería más sencilla de realizar.

Durante su desarrollo nos encontramos con el problema de que no sabíamos bien cómo gestionar el tema de los certificados, así que decidimos dejarlo aparte para realizarlo más adelante.

Para la segunda entrega teníamos que realizar la parte del front-office, que se centraba en la parte del cliente, como sería la reserva de plazas para una actividad, así como el entorno de usuario. Aunque para realizar toda esta parte debíamos haber terminado cierta parte de los monitores, que sería la creación de actividades, puesto que si no hay actividades no se puede reservar ninguna. Una vez terminado esto desarrollamos la parte de la reserva de actividades, centrándonos más en un diseño práctico y funcional antes que en uno estético por falta de tiempo.

A pesar de haber desarrollado la parte de las reservas nos faltaban cosas, como la interacción del sistema con el usuario o el mostrar las reservas. Eran pequeños detalles que marcaban una gran diferencia funcional, pero que acabamos solucionando los problemas que teníamos y pudimos implementarlo.

Para la última entrega debíamos desarrollar la parte estética del proyecto, pero no solo eso, sino también terminar lo que nos faltaba en las partes de los monitores y de los clientes y desarrollar toda la parte del administrador del sistema. En la parte de los monitores quedaba gestionar el tema de las solicitudes y de los certificados que habíamos dejado para más adelante. En la parte de los clientes faltaba hacer un sistema funcional e interactivo con el usuario.

5. Mejoras

En cuanto a mejoras, como nuestro grupo estaba formado por solo tres personas, no era necesario realizar ninguna de las mejoras propuestas.

6. Conclusiones y trabajo futuro

Este trabajo nos ha servido para tener una idea de cómo realizar un proyecto en la vida real, qué técnicas usar y qué herramientas utilizar. Además, hemos usado los conocimientos de otras asignaturas para el trabajo, como base de datos, que es algo que nos ha gustado, así parece que las asignaturas no son completamente independiente, si no que vemos cómo se relacionan unas con otras.

Además, hemos aprendido la importancia del uso de patrones de diseño para simplificar el trabajo a realizar y ofrecer una solución estandarizada a un problema conocido, de manera que su implementación y posibles cambios futuros sean más sencillos.

7. Bibliografía

Para terminar, la bibliografía empleada ha sido la recomendada desde la asignatura, y en caso de no ser suficiente, se ha buscado en internet información complementaria. Para la base de datos se ha seguido el libro de Merche Marqués, que junto a los conocimientos adquiridos en la asignatura EI1020 Bases de Datos, han sido suficientes y no ha hecho falta buscar más información.