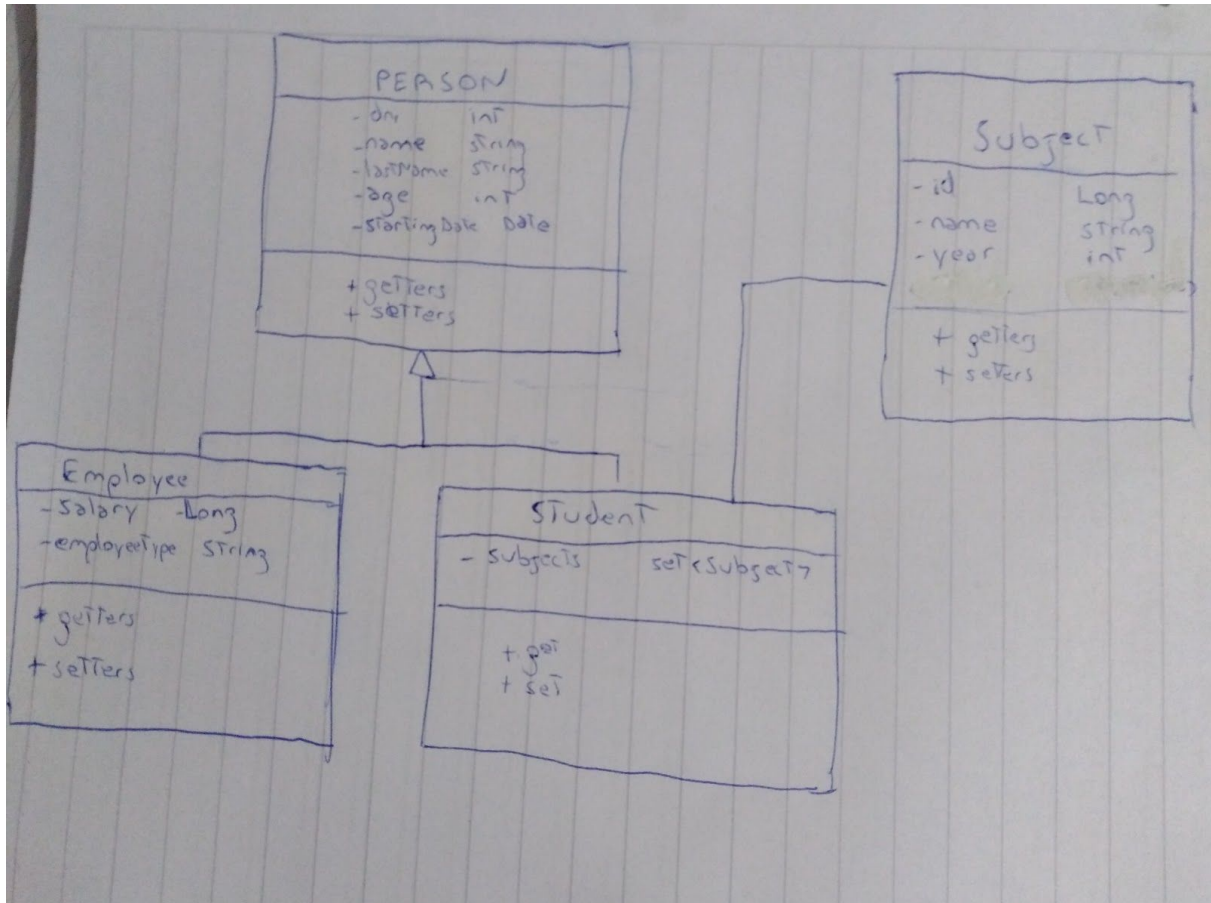


- 1) Create a domain model implementing the concepts of hierarchy, having as example a school's structure where we find different types of people. Examples of possible entities are: Principal, Employee, Student and Janitor. Try to implement the concepts of Abstract class and Interface.

Diagrama de clases:

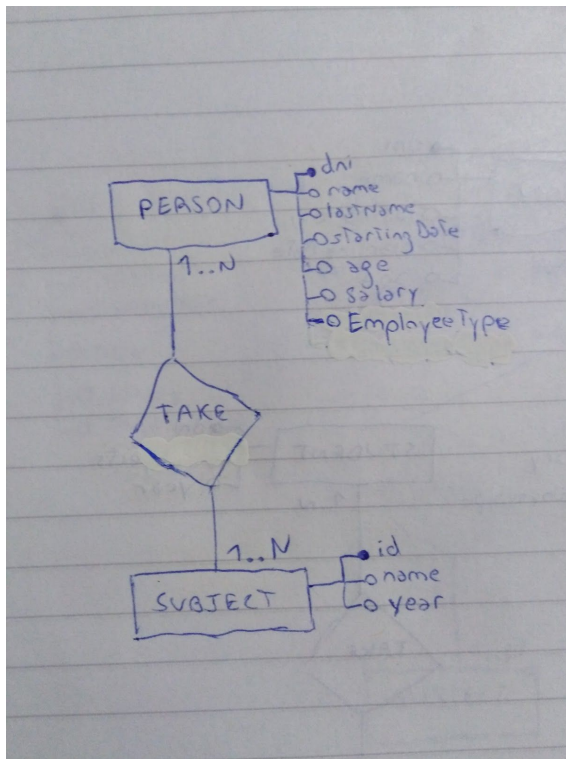


Aclaración: decidí no crear las subclases Janitor y Principal ya que el enunciado no hace referencia a que tengan alguna responsabilidad o comportamiento diferente, por lo tanto, en el caso de implementarlas tendría que crear dos clases igual, lo cual no tiene sentido. Por este motivo solo agregue el atributo `employeeType` en la clase **Employee** para diferenciar los puestos de trabajo de los empleados.

En el supuesto caso de que Janitor y Principal tuviesen distintos atributos o responsabilidades diferentes, se deberían crear las correspondientes subclases que hereden de la clase **Employee**.

C) Create different database structure diagrams with the possible implementations of the created domain.

Opción 1: Toda la jerarquía en una misma tabla. (Single table)



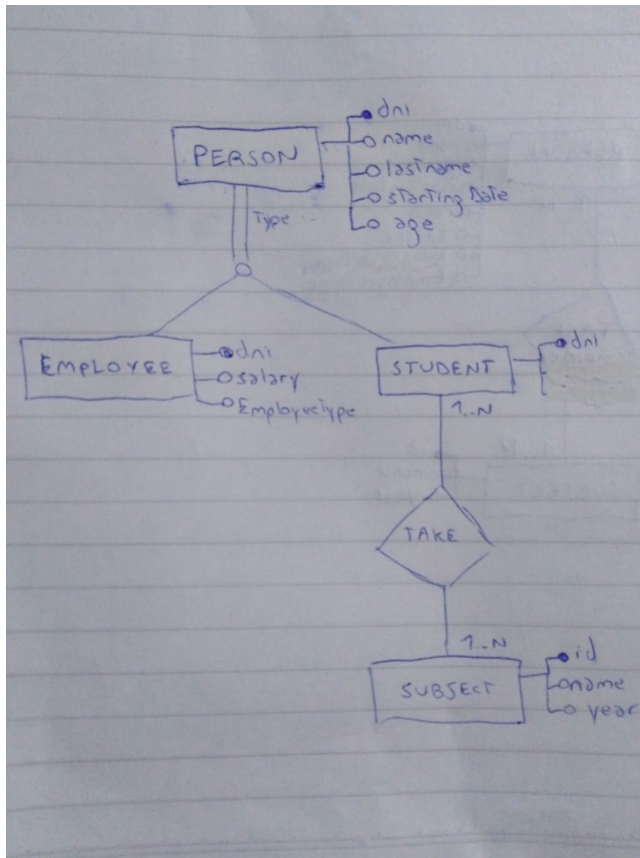
Ventajas:

- Si se hacen consultas sobre todas las personas esta opción es muy eficiente ya que toda la información está en una sola tabla y no es necesario hacer joins.
- Cuando se agrega un atributo a la clase padre, el mapeo es muy sencillo ya que solo basta con agregar la columna a la única tabla existente.

Desventajas:

- La tabla va a contener tuplas con columnas vacías ya que no todas las clases de la jerarquía poseen los mismos atributos, por ejemplo, en el caso de un estudiante, la columna salario estará siempre vacía.
- Si se desea hacer una consulta sobre una subclase, por ejemplo, recuperar los estudiantes, se van a recorrer filas demás en la tabla, ya que también se recorrerán los empleados.
- Si las clases comparten un atributo que posee distinta semántica o restricciones para cada subclase, se deberá agregar una columna nueva por cada clase.

Opción 2: Una tabla por clase. (Table per class)



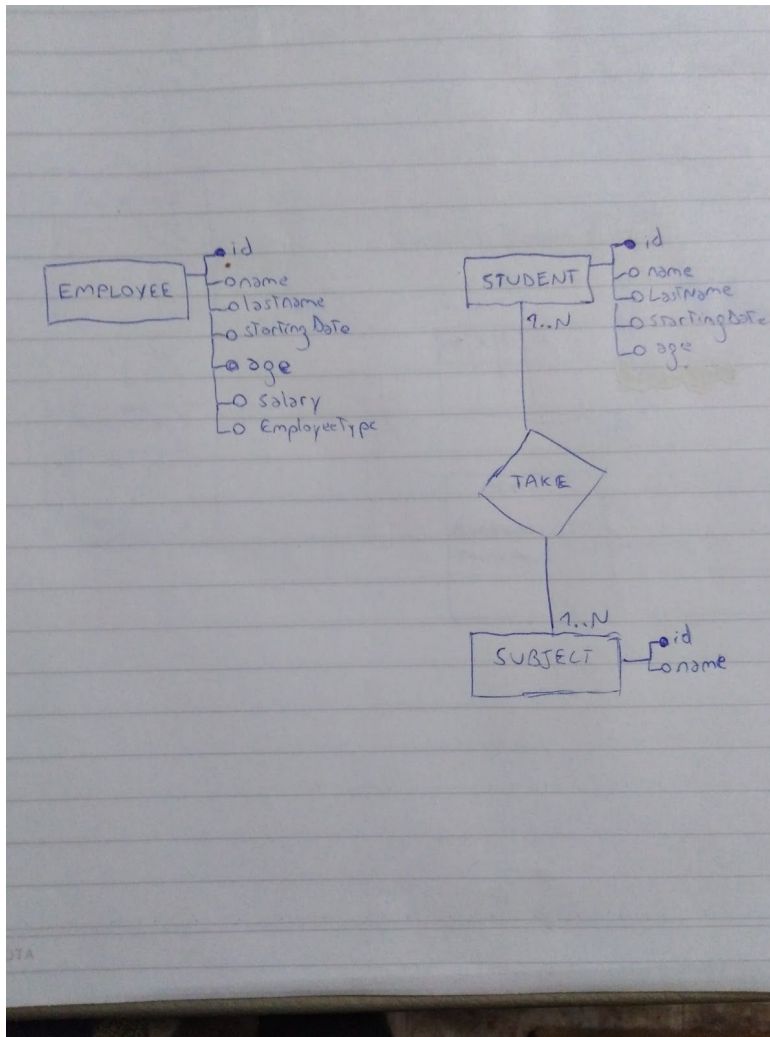
Ventajas:

- Se tiene una tabla que representa la clase abstracta con los atributos en común y tablas para las clases concretas que apuntan a la tabla de la clase abstracta.
- Si se agrega un atributo en común a todas las subclases, solo se agrega una columna a la tabla de la clase abstracta.
- Si se quiere hacer una consulta sobre los estudiantes, se recorren las filas necesarias, ya que la información está en la tabla estudiante. (En el caso que se necesiten atributos de la tabla persona habrá que hacer un join).
- Si se quiere hacer una consulta sobre todas las personas, basta con hacer una consulta a la tabla persona (en el caso de que posea los atributos necesarios a retornar).

Desventajas:

- La mayor desventaja es que cuando se tienen que obtener las instancias completas de una clase siempre hay que hacer joins entre tablas para obtener los atributos necesarios. Ejemplo para obtener todos los estudiantes, hay que hacer join con la tabla persona para obtener dni, nombre, etc.

Opción 3: Una tabla por clase concreta. (Mapped Superclass)



Ventajas:

- Para recuperar la información de una instancia basta que con devolver la tupla correspondiente en la tabla. No hay que hacer joins ya todos los datos están en esa tabla.
- Si se quieren obtener todos los estudiantes, no se recorren filas demás, todo está en la tabla estudiante. (No se recorren los empleados).

Desventajas:

- Si se quieren obtener todas las personas hay que hacer una unión entre las tablas.
- Los atributos en común son repetidos en todas las tablas, por lo que si se quiere agregar un nuevo atributo en común hay que agregarlo en cada una de las tablas.

d) Consider that I want to query all the people in the database and the engine is taking too long to respond. Which are the possible solutions? How would you determine the problem?

Para esta consulta haber elegido la opción *Single Table* es lo más conveniente, ya que la información a devolver es la tabla *Persona* completa, por lo que en este caso no es necesario hacer joins entre tablas (*Table per class*) ni hacer uniones (*Mapped Superclass*).

Si la query para obtener todas las personas está tardando mucho tiempo en responder, una buena solución sería utilizar Paginación. Esto reduce considerablemente el tiempo de consulta ya que se obtiene la información en pequeños grupos de datos.

Spring Data nos provee todo el soporte para implementar paginación en nuestra aplicación de una manera sencilla.

e) I want to query all the students which ages are between 19 and 22. How would you optimize this query?

- Si se utilizó la estrategia *Single Table*, podemos optimizar esta consulta creando un índice multinivel. El primer nivel debería ser por el campo *tipo_persona* y en el segundo nivel podemos tener un índice por rangos de edad.
- Si se eligió la estrategia *Table per class*, se puede hacer ordenar la tabla de estudiantes por edad y crear un índice de agrupamiento por edad, teniendo rango de 3 años de diferencia.
- En el caso de la opción *Mapped superclass* podemos utilizar la solución anterior. (Índice de agrupamiento)

F) I want to delegate the saving process to the database engine instead of having it on the application side, which database utility would you use?

Usaría *Store Procedures* para ganar en términos de performance ya que no hay comunicación extra entre la base de datos y la aplicación y a su vez no es necesario compilarlos por cada instancia ya que los *Store Procedures* se compilan solo una vez.

Con esta solución también se puede reducir el tráfico de la red entre el cliente y el servidor ya que los comandos se ejecutan como un único batch de código. Esto significa que solo la llamada para ejecutar el procedimiento se envía a través de una red, en lugar de que cada línea de código se envíe individualmente.

A su vez proporcionan una importante capa de seguridad entre la interfaz de usuario y la base de datos, ya que los usuarios finales pueden ingresar o cambiar datos, pero no escriben *Store Procedures*.

2) Create the domain model of a conference room schedule where meetings are booked. Bookings should contain a list of participants and only the organizer can modify them.

Incorporate all attributes and methods you consider needed for each domain entity. Have into account that bookings cannot overlap and cannot last less than 15 minutes and no longer than 3 hours.

Create a diagram of how the user interface would look like and how a booking flow should be

Diagrama de clases:

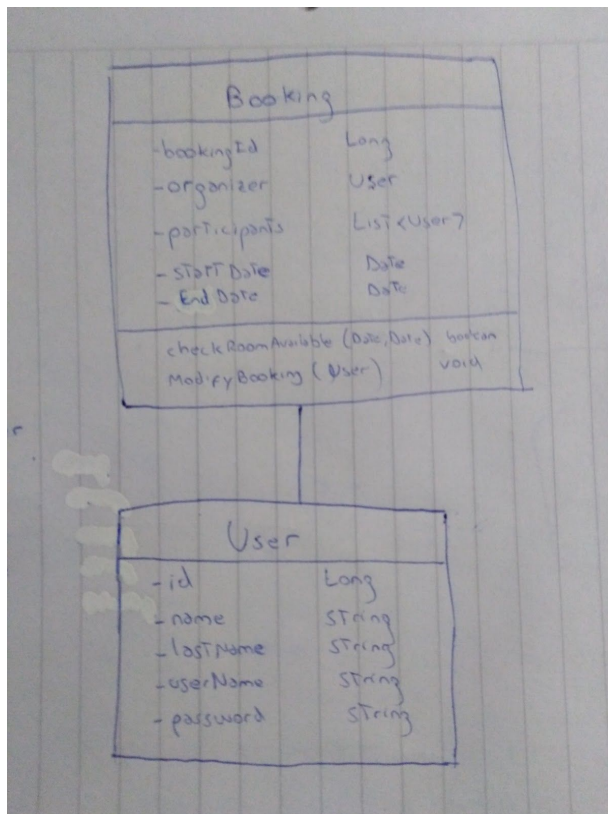
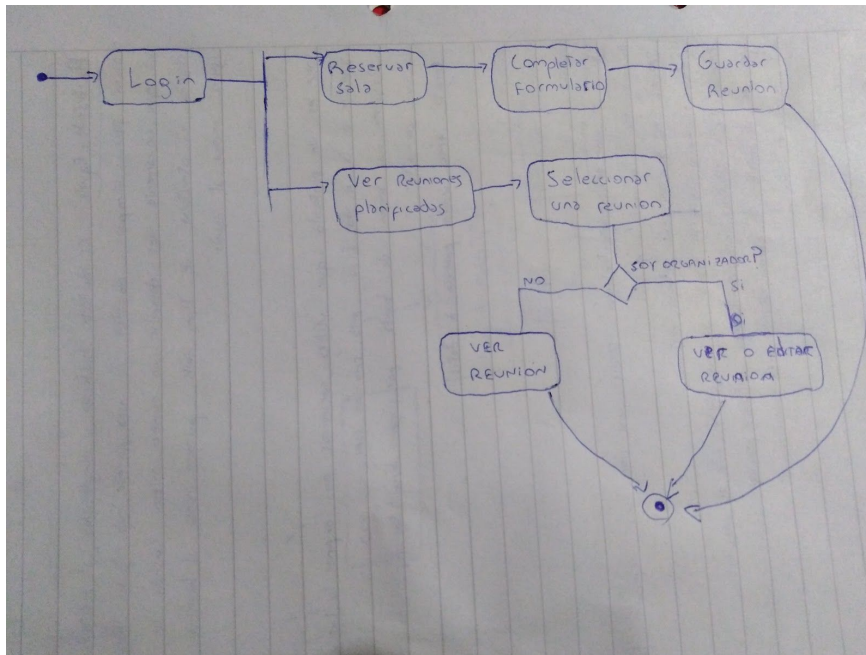


Diagrama de flujo:



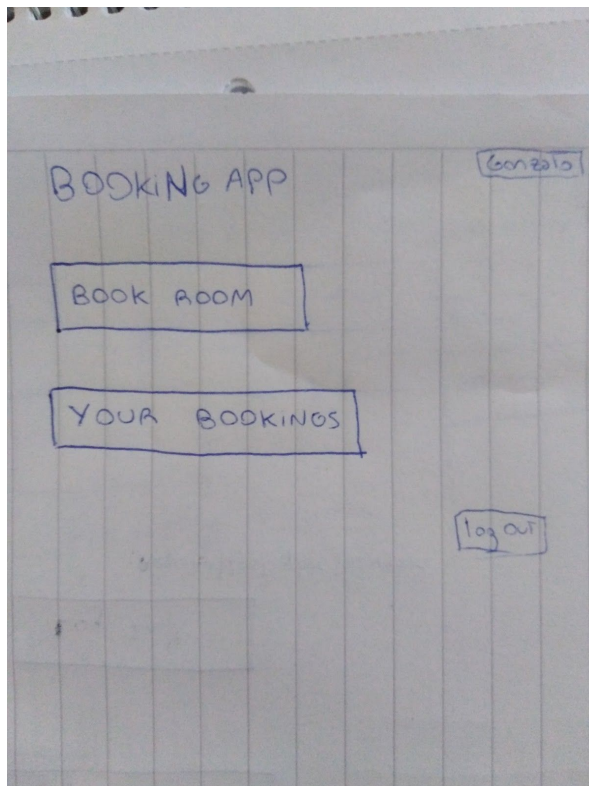
Interfaz de usuario

Login page: La primer imagen hace referencia a la página de login de la aplicación.

A hand-drawn sketch of a login page on lined paper. At the top, the text "BOOKING APP" is written in blue ink. Below it, the label "Username" is followed by a rectangular input box containing the text "admin". Further down, the label "Password" is followed by a rectangular input box containing five dots. Below the password box is a rectangular button labeled "Log In". At the bottom, there is a link that reads ". Register / Lost your Password?".

Home Page:

Luego de que el usuario se logue en la aplicación accederá a la siguiente vista donde habrá un botón para consultar las reuniones ya planificadas y otro botón para reservar la sala.



Book Room Page:

Luego de clicar el botón *Book Room* se accederá a la siguiente pantalla donde se deberán completar los campos correspondientes de la reserva.

Entre otros campos podemos ver que se debe seleccionar una fecha, para eso se muestra un calendario donde se podrá seleccionar una fecha en que la sala esté disponible. Una vez seleccionada la fecha se deberá elegir un horario de inicio y fin donde la sala esté libre. Con las horas ingresadas se hará el chequeo de que el tiempo de duración de la reunión sea correcto, en el caso de que exceda los límites se mostrarán los correspondientes mensajes de error.

También se le puede agregar un título y descripción a la reunión y vemos un campo donde se puede invitar a los participantes que tienen que asistir a la reunión.

Book Room

Gonzalo

Title

Description

Date

From To

Mon	Tue	Wed	Thu	Fri	Sat	Sun
30	31	1	2	3	4	5
6	7	8	9	10	11	12
...						

Participants

Gonzalo ☒

Martin ☐

Nicolas ☒

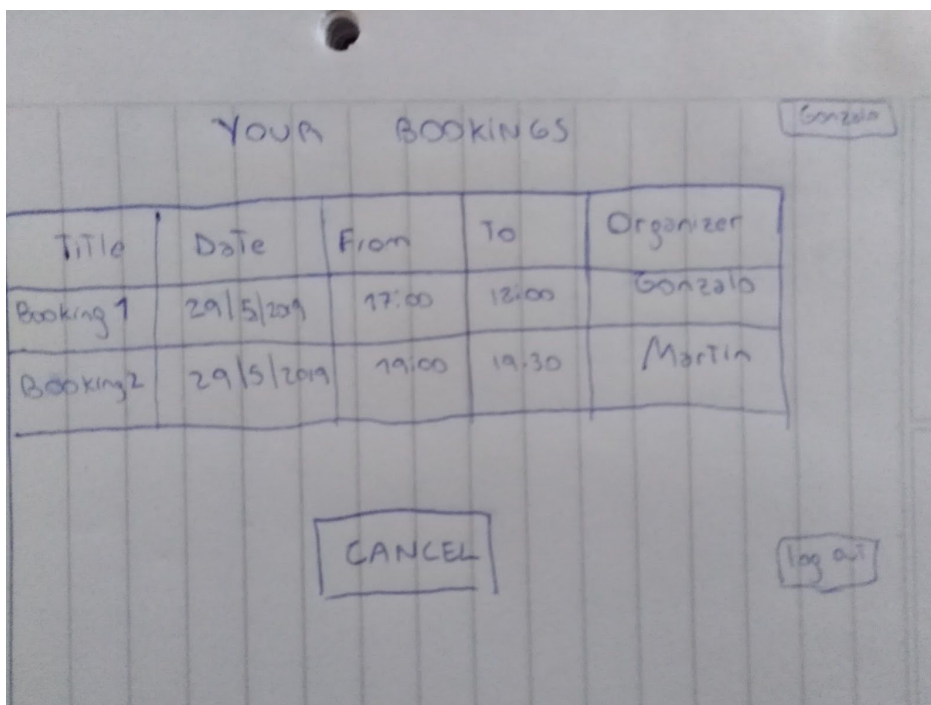
CANCEL

SAVE

Log Out

Your Bookings Page:

En el caso de clicar el botón *Your Bookings* desde la homepage, se redireccionará a la siguiente vista donde se podrá ver en una tabla las reuniones a las que tiene que asistir el usuario. En el caso de que sea Organizador de una de ellas podrá clicar la correspondiente fila y será redirigido a una vista similar a la anterior para poder editar la reunión. De lo contrario solo podrá ver la información de la reunión.



Title	Date	From	To	Organizer
Booking 1	29/5/2019	17:00	18:00	Gonzalo
Booking 2	29/5/2019	19:00	19:30	Martin