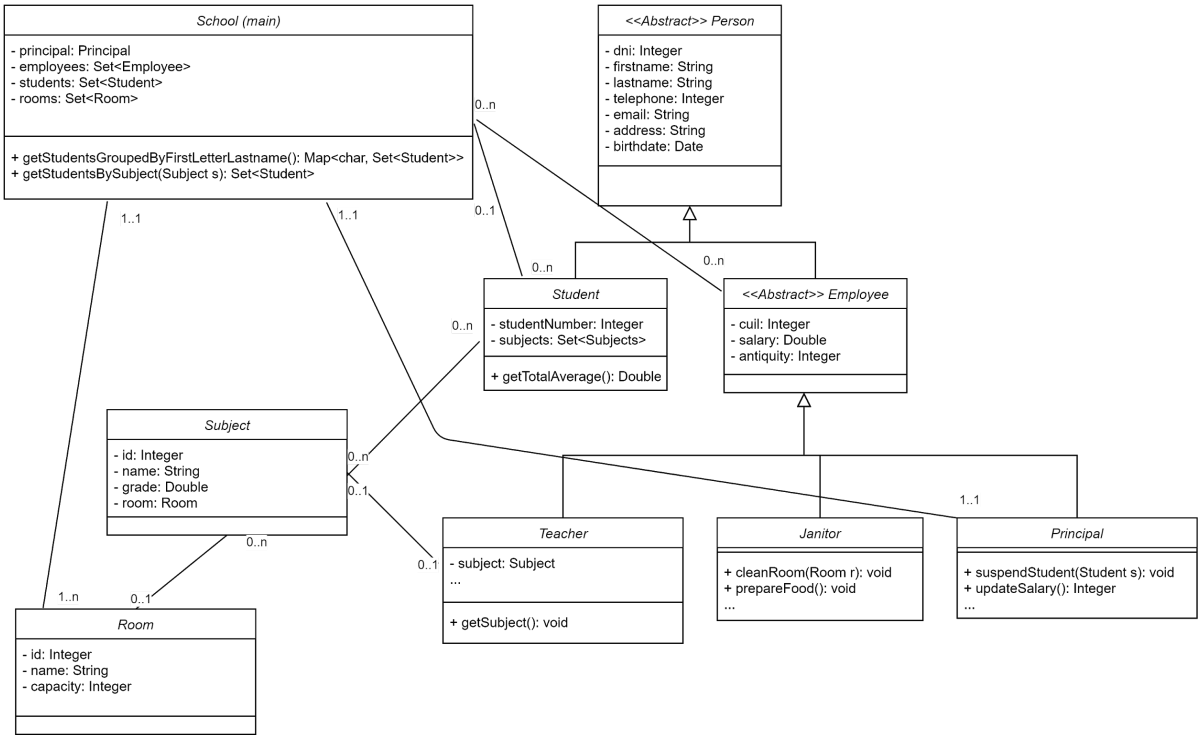
Teracode: Developer hiring exam
Exercise One

A) Considering the structure of a school, create the diagram that describes the domain model and relationship between entities. Some of the possible entities may be Principal, Employee, Student and Janitor. Try to apply the concepts of Hierarchy, Abstract Class and Interface

Applying the concepts of object oriented programming, this is the result diagram:



This diagram is based in java, basic **getters and setters** are not included for better readability. In case of using a framework like spring or using a database and ORM such as hibernate, this diagram could change to fulfill the specifications of the tool.

B) Implement a Java method that returns all the students in the school grouped by the first letter of their last name.

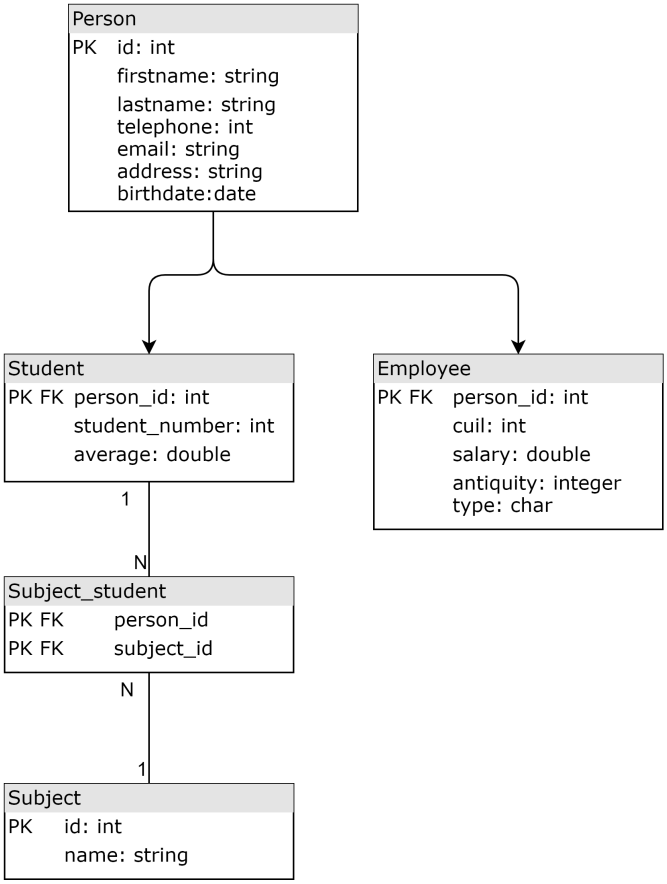
The solution is getStudentsGroupedMyLastnameFirstLetter in com.application.service.SchoolService in the repository POO

C) Implement a java method that returns all students taking a specific subject. Please consider that the students cannot be repeated, this needs to be verified.

The solution getStudentsBySubject in com.application.service.SchoolService in the repository POO. A set it's used, this means that the repetition of student is validated implicitly with the definition of equal in the model of Student.

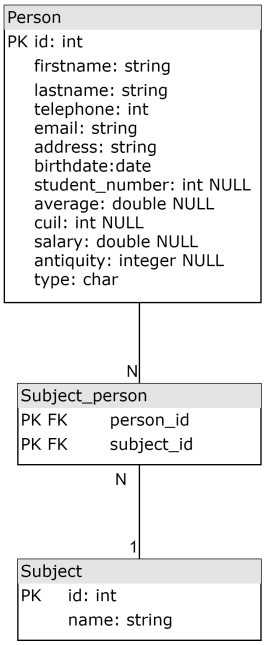
D) Create at least two different database table structure diagrams to describe the possible ways that the domain could be translated to the database. Explain pros and cons of each structure.

Database schema 1



Using a schema with type hierarchy and the relationship student-subject normalized, the system could be more scalable and maintainable, allowing to add relationships and entities that does not affect the existing models and services. The cons of using this schema is that the services could not have the best performance and be more complex, because it is necessary to perform joins between tables for some queries.

Database schema 2



In a schema with a discriminator column (type) in the table person, the performance of select queries could improve respect to the first schema and also, to add another person type only requires to add a new value in that column, making it more flexible and simple. The cons are that if the new person type has other fields, it could create a conflict with others services using the same table. Also, this field will be available for other types of persons making it less readable and inconsistent with the domain models, creating the necessity to perform more validations.

E) Having the following query:
SELECT * FROM janitor j
INNER JOIN employee e ON e.id = j.id
INNER JOIN person p ON p.id = j.id
WHERE j.workingArea = 'Hallway';
Our query is taking too long to respond. What changes would you do to the query and/or the database to make it go faster if we only need the first and last name of the janitor?
For a better performance, it is recommended to only retrieve the necessary columns (p.firstname, p.lastname) and not all the columns from all the tables in the query (*). In this case, it is not necessary to perform the operation "INNER JOIN employee e ON e.id = j.id" if the id is the same in the three tables and it can't be null if it is the primary key. Also it is a good practice if the restriction j.workingArea = 'Hallway' is made first in a subquery.
The result query is the next:

```
SELECT p.firstname, p.lastname FROM person p
WHERE p.id in
    (select j.id
    from janitor j
    where j.workingArea = 'Hallway');
```

Creating a concurrent index in the column workingArea from the table Janitor could also improve the performance of the query.

F) Consider that we have a query that joins many tables and takes too long to return the values. We know as a fact that the tables involved do not update too often, but we still use this query many times in a day (consider it a report). What would you do to be able to get these results faster?
In this case that the involved tables are not often updated it is recommended to use a materialized view to get the results already processed and saved in a cache table because it has to be recalculated much fewer times.

G) Using the model create a query that returns all the students with age between 19 and 21. Take into account that the age is not a column on the table, we have a field for birth date. How would you optimize this query?

```
select * from student s
inner join person p on s.id = p.id
where p.birthdate between
current_date - interval '21 year' and current_date - interval '19 year'
```

In case that the only information needed is from the table student, the query can be improved as in exercice E:

```
      select s.* from student s
      where s.id in
          (select p.id
          from person p
          where p.birthdate between
           current_date - interval '21 year' and current_date - interval '19
year')
```

A field “age” calculated with a trigger in the table person could be created or an index in the column birthdate in the table person could optimize this queries.

H) We need to build a new application and we want to be able to have the business logic on the database engine instead of having it in our Java code. Can you suggest a way of doing this? As an example, how would you persist a student? What are the pros and cons of this?

This can be achieved using stored procedures, triggers and views. For example if the repository layer saves a student, without knowing that some data has to be persisted in the table person and other in the table student, it can be done in the database using a trigger “instead of”, which intercepts the operation and saves the information correctly.

For example if we have a view with all the information about the students (person + student data) the view and the trigger to insert an student are:

```
create or replace view student_view as
select *
from student s
inner join person p on p.id = s.id;
```

```
create or replace function fn_insert_student() returns trigger as $$
begin
    insert into Person values(new.id, new.firstname, new.birthdate, ...);
    insert into Student values(new.id, new.student_number, new.average);
    return new;
END
$$ language plpgsql;

create trigger tr_insert_student
instead of insert
on student_view
for each row
execute function fn_insert_student();
```

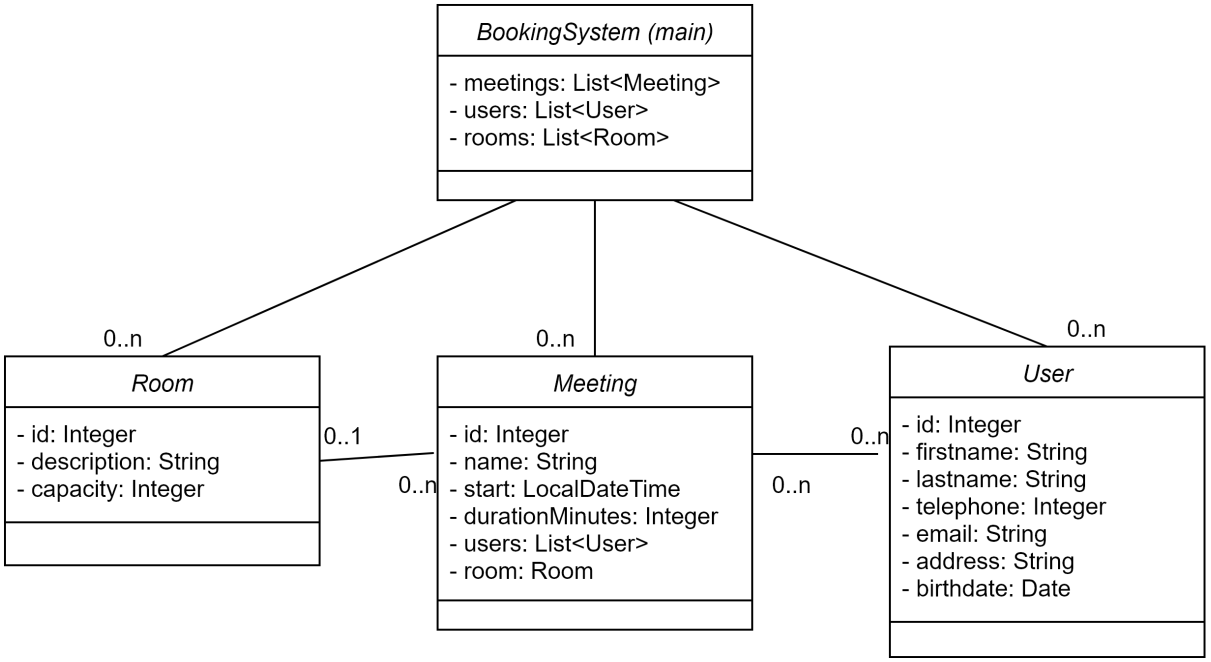
As pros, this avoids code duplication, simplifies the java code of the application and allows the same operations to be useful between multiple applications that use the same database.

As cons, this depends on the DBMS implementation, it needs more specific knowledge and it is more complex to maintain, to migrate, to test and to debug.

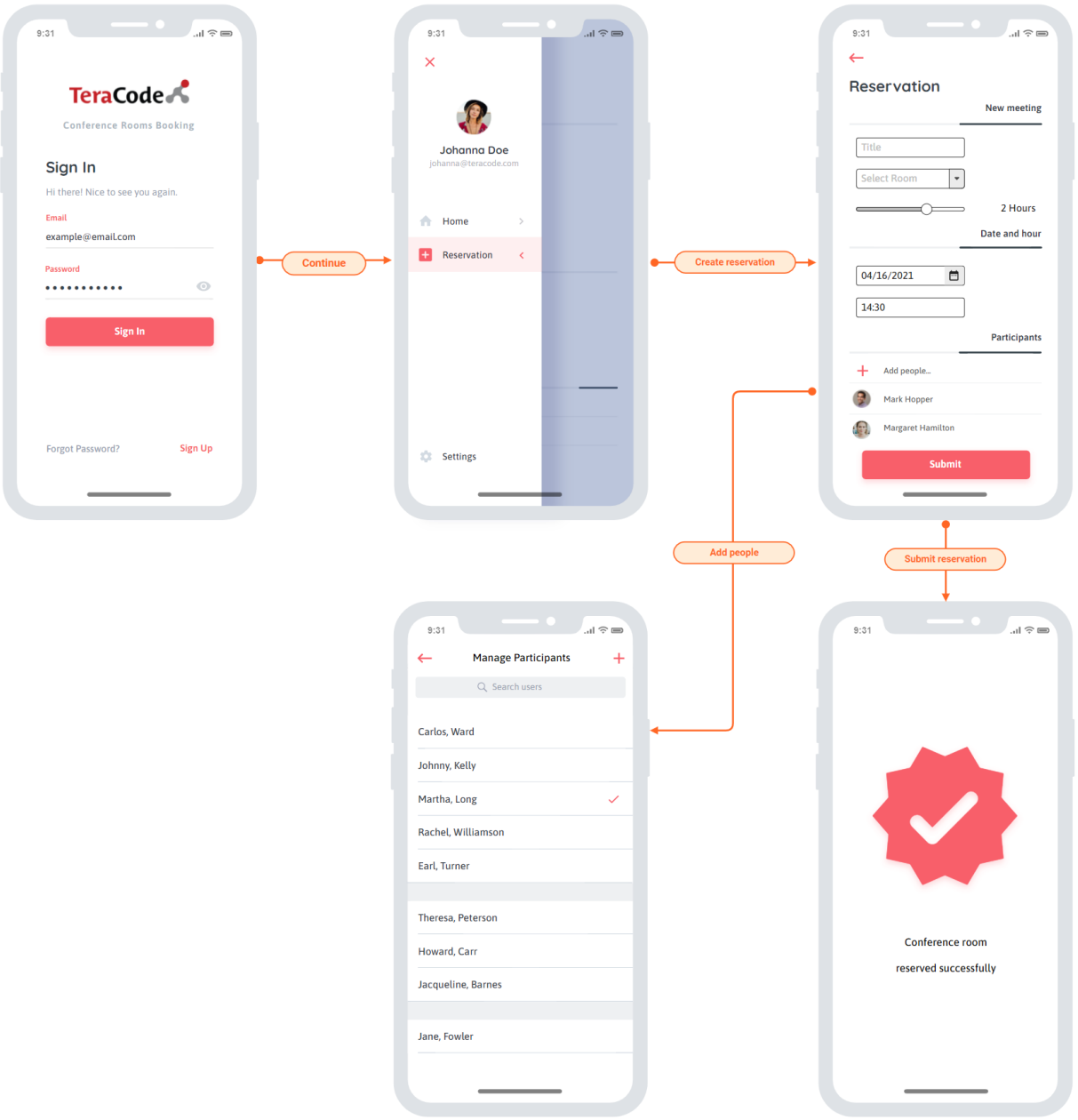
Exercise Two

We need to create a Conference Rooms Booking application, knowing this:

A) Create the domain model diagram with all the possible entities, its attributes and relationships



B) Deliver diagrams of the user interface for each step of a booking process.



https://app.moqups.com/C7pWfiNhuR/view/page/ae8fe8eb0

C) Considering that meetings cannot last less than 15 minutes and more than 3 hours, also meetings cannot overlap with each other in the same room, deliver the validation method that is in charge to verify these rules.

The solution is in com.application.service.Booking Service in the repository POO