Rest API

All of these methods have not been implemented, and rather serve as theorised methods that we wanted to try and implement.

Methods	Description
GetAllZipcodes	This would return all the zipcodes in Denmark.
GetPerson	This returns everything about a person based on their ID, or in the case of a similiar function, their Phone number.
GetPeople	This returns every single person in the database.
GetPeopleFromZip	This returns every person that has a certain zipcode.
AddPerson	This naturally would add a person.
EditPerson	This would edit a person based on an input person and an existing person's id.
DeletePerson	This deletes the person.
GetByHobby	This gets people that have a certain hobby. Not even in the facade does this function properly.
GetCompany	There's three variations of this, one by ID, one by Phone number and one by CVR.
GetCompanies	This gets all the companies.
AddCompany	This adds a company.
EditCompany	This would edit a company.
DeleteCompany	This deletes the company.

Error Responses

Because of problems with the API, we never got around to making any sort of error handling, so there's no responses that can be sent out since there's no error to perceive.

JSON format

We do have a JSON converter that would be able to convert JSON data to entities, but when it began to bug out we never really found a way to fix it so we didn't get around to doing the other way around.

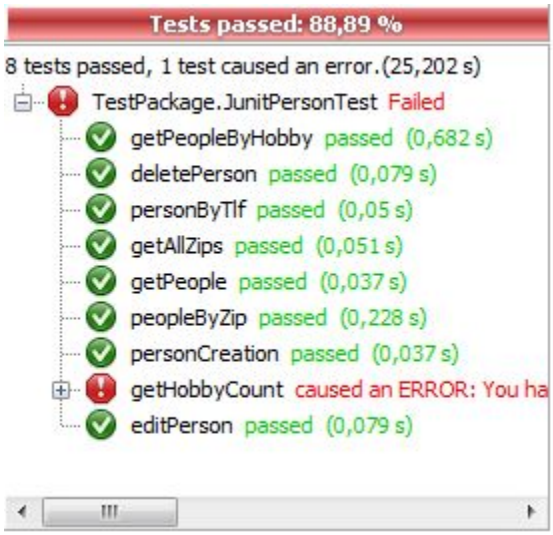
Test strategies and results

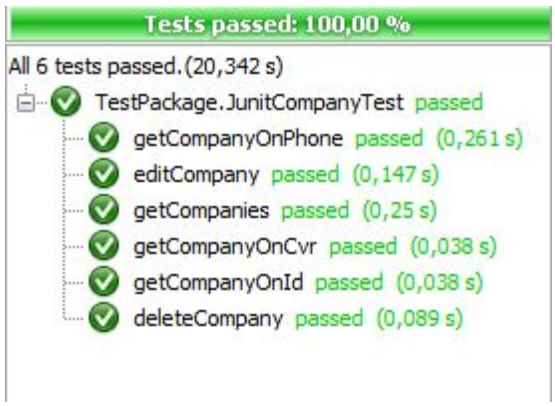
Unit Tests

Unit tests made up most of the tests that were performed on the system, primarily regarding the backend methods in the facade. By steadily going through each method to try and fetch data from a spare database, we were able to determine if the various methods worked.

Facade Tests

Testing the Facades was no easy feat, as the many tests revealed that our JPA syntax would continually throw out errors in terms of either being unable to properly retrieve from the database, or just being plainly written wrong and throwing an error before it even began to access it. Though that's part of the reason why we do so many tests, so that we have the ability to correct the broken code so it wouldn't become a problem for the front end. We had two separate facades and relevant tests, one for persons and one for companies.





API Tests

The API ended up malfunctioning and not being usable, so we've been unable to actually test it. A shame, really.

Work Delegation

Most of the work on the Facade was done by Joachim, with some help from Martin. Since the facade functions as our connection to the database, we suggest they get the full 5 points for contributing to the project.

The work done on the REST-API was handled by Nikolai and Peter, and despite the problems that arose with it and how we ended up not being able to use it, they deserve the 5 points available for doing their best to make it work.

Inheritance Integration

We've chosen to use Table Per Class type inheritance, as we thought it was the best way to have both a company and a person class that's still able to have their own emails, their own phones, addresses and so on. If we only had to worry about one class, we probably would've gone with a single table, but this is how it ended up.

How to Test the system

You kinda can't :/. If we had more time to figure out how to make the API work, we would've integrated some dummy data with a SQL script, at least to allow people to search for preset values, but we just couldn't manage it in time.