



Apartments.com

Made by group 6:

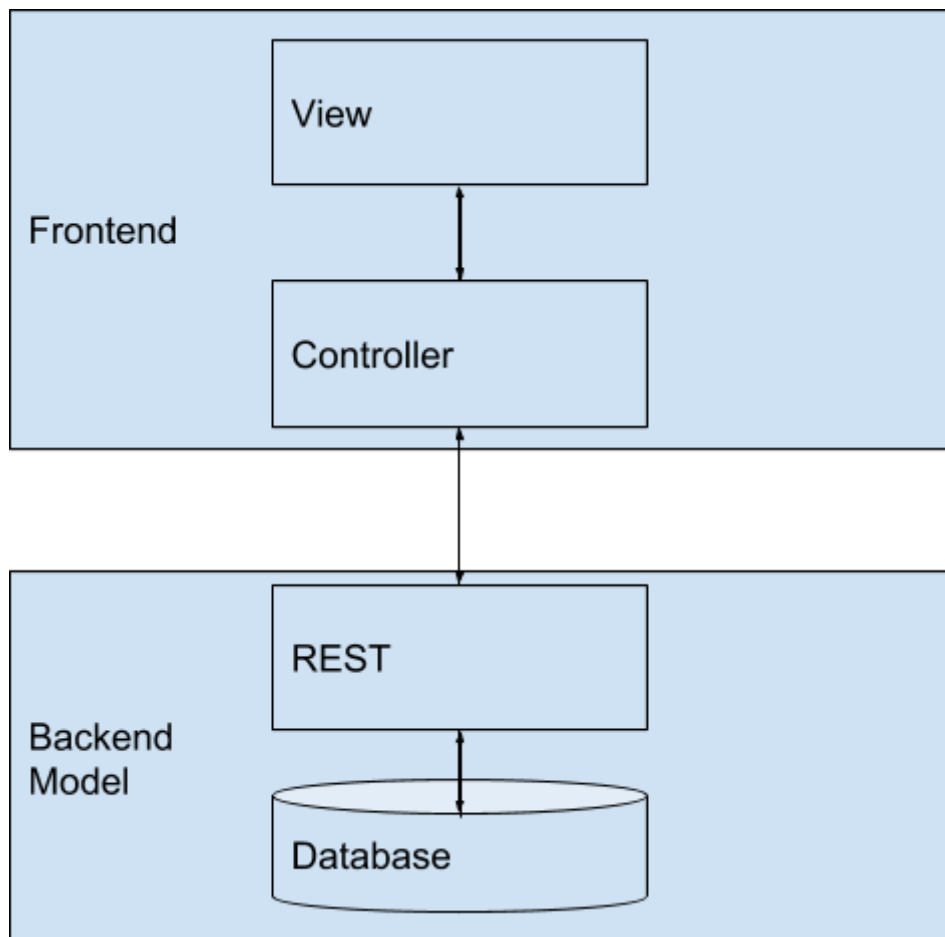
Emilia Vestlund, vemlia@student.chalmers.se, 19960130-9363
Therese Stuesson, thestu@student.chalmers.se 19960708-2667
Amanda Jonsson, amajon@student.chalmers.se, 19950521-4784

Content

Architecture	3
Design	3
The used techniques	4
Flow	5
Testing	5
Use Cases	5
Add	5
See	5
Update	5
Delete	6
Search	6
Help/contact	6
Request	6
Other	6
The responsibilities	6
Thereses responsibilities	6
Emilias responsibilities	7
Amandas responsibilities	7
Picture on main page	8
Picture of the group	9

Architecture

We used MVC in our web application. The View is the html pages, the model is the database and the controller is the connection between the view and the model which is located in the frontend of our project. We have a full stack application with a JEE Rest-backend and a JS frontend. The architecture is represented by the picture below.



Design

We have the following html pages in our web application: index.html, rent.html, addRent.html, rentOut.html, addRentOut.html, contact.html, help.html.

We are using the same layout in the header in all pages besides the home page, to get consistency. The index.html represent our home page. We have an slideshow on the home page. We use the same layout in both Rent and Rent Out. In the index.html, help.html and contact.html we use the two page grid class to structure the page.

When you enter the web application you reach the home page. There you have two options: either you can click on the pictures that says: apartment wanted or for rent or you can use the navigation bar at the top of the page. The website is fully connected which means that all

pages link to all other pages and they have a global navigation feature, which in our case is the navigation bar. In Rent and Rent Out you can see a table of advertisement and you can click on a button that says "Add a new ad". If you click at that button you will be navigated to a new page where you can fill in some information about what you want to rent or rent out and when you press the "Add" button the ad will be added to the table and the user will be navigated back to the Rent or Rent Out page.

We used some external services for CSS framework, bootstrap and W3Schools. We used the Bootstrap framework on the the forms in addRent.html and addRentOut.html because it was an efficient way to get a good design and it was easy to use. We also used it on the pop-up window since we were unsure how to design a pop-up window. In our slideshow on the indexpage we used the CSS framework W3Schools because we did not thought it was important do our own css for a slideshow, when it would look about the same. We did do the function carousel() in javascript for the slideshow on our own and chose that the pictures should show 5 seconds each.

The rest of the CSS we did ourself and we have different CSS classes so it will be easy to get an good overview of the CSS. We are interested in user interfaces so it was fun to do the design from scratch so we focused a bit on the CSS. For our google map in the contact.html we used googles API to connect to the map. In the map we have marker which is a gui component for the google map. We have used several GUI components for example buttons, forms, labels, fieldset in for example the contact form.

The used techniques

We used NetBeans as the develop environment and Glassfish 5.0 as the server because they were easy to use together. We used the persistence techniques to be able to use the Entity in the core files. We use rest in the backend and javascript in the frontend. We used AJAX because it was easy to connect the frontend and the backend this way. In the rest backend we use http-requests: post, get, put and delete.

We have validation on the email input in the forms in addRent and addRentOut to make sure the user fill in an email address. Then we have required on some of the inputs in the form that we think a user must fill in to make sure the most important information get published. On the contact form we have required on all input since we believe we need all the information to be able to help the user.

We discussed the use of exception handling and came to the conclusion that we would like to have exception handling on the backend. But we struggled with the backend the two first weeks and had to spend a lot of time working on that. We had started on a backend but we realised that it was not testable so we rethought and started on a new backend. Due to this we did not have the time to implement exception handling.

We wanted to have the functions registry and login on the site so that a user only would be able to update and delete their own ads. But since we were only three persons on the project

and had so much problem with the backend in the start we did not have time to implement these functions.

Flow

An example on the flow in the application is when you add a new ad on the webpage rent. To add a new ad in rent you have to go from the index site to the rent site and click on the button "Add a new ad". This way you get to the right site. When you have filled in the form and click on the button "Add" the listener in the controller in the frontend (RentCtrl.js) will be notified and the data will be sent to the create method in RentCtrl. From this method the data will be sent to the create method in the frontend model (RentRegistry.js). When this method return later the observer will be notified again. From RentRegistry the data will be sent to the service in the frontend (RentService.js) to the method create. This file have the connection to the backend (the url for the backend) and from the method the data will be sent to the rest in the backend (RentResource.java). RentResource then sends the data to RentCollection that uses the entity manager to add the data to the database.

Testing

We tested the backend with curl tests. With this kind of test we were able to test all the methods we had (find, find all, count, create, update, delete) in the terminal to see if the methods worked.

We did not have a specific method to test the frontend. Instead we used the browser to inspect the sites and mostly used the console to troubleshoot the sites when something were not working.

Use Cases

Add

As a user I want to be able to add an ad so that I can rent out an apartment.

As a user I want to be able to add an ad so that I can rent an apartment.

See

As a user I want to be able to see a list of all the apartments I can rent so I can see my options.

As a user I want to be able to see a list of all the apartments I rent out so I can see my options.

Update

As an apartment owner I want to update my add if something change.

As a user I want to update my add if something change.

Delete

As an apartment owner I want to delete an add so it will disappear when it is rented.

As a user I want to delete an add so it will disappear when I have found an apartment to rent.

Search

As a user I want to search for an specific add, so I will find adds quick.

Help/contact

As a user I want to find the company with an map.

As a user I want to send question to the company with a questionnaire.

As a user I want to be able to get help on the website.

Request

As a user I want to be able to enrol interests for apartments.

As a user I want to be able to message possible renters.

Other

As a user I want it to be easy to navigate on the website.

Useccases we thought about but did not have time for:

As a user I want to be able to register on the website.

As a user I want to be able to login on the website so I can only update and delete my ad.

The responsibilities

Thereses responsibilities

The backend:

Adding all the database files (beans.xml, glassfish-resources.xml, glassfish-web.xml web.xml, persistence.xml)

The files for the rent out table (RentOut.java, RentOutCollection.java, RentOutResource.java)

Testing the methods for the rent out table with curl

The frontend:

The controller for rent out (RentOutCtrl.js)

The model for rent out (RentOut.js, RentOutRegistry.js)

The service for rent out (RentOutService.js)

The view for the rent out page (RentOut.html, addRentOut.html, RentOuttable.js)

Emilias responsibilities

The backend:

The file DatabaseResource.java, DataSupplier.java

The standard config files (ApplicationConfig.java, CORSFilter.java)

The abstract classes (AbstractDAO.java, AbstractQuery.java)

The frontend:

Adding observers (eventBus.js)

CSS pages (style.css, contactcss.css, helpcss.css, indexcss.css)

All other html pages (contact.html, help.html, index.html)

Amandas responsibilities

The backend:

The files for the rent table (Rent.java, RentCollection.java, RentResource.java)

Testing the methods for the rent table with curl

The frontend:

The controller for rent (RentCtrl.js)


The model for rent (Rent.js, RentRegistry.js)

The service for rent (RentService.js)

The view for the rent page (Rent.html, addRent.html, Renttable.js)

Picture on main page


[Home](#) [Rent](#) [Rent out](#) [Contact](#) [Help](#)




Grip Stamato

Welcome

Welcome to apartments.apartments.com. This is a site for you to find apartments to rent our rent out your apartment on.





Apartments.com is developed by: Amanda Jonsson, Emilia Vestlund and Therese Stureson.

Contact information: apartments@apartments.com

Picture of the group



From the left: Amanda, Emilia and Therese