**Documentation Sprint B**

**ARQSI**

Patryk Pas

Juuso Jalava

Miika Kauppinen

Benjamin Moreira

Liam Saliën

Contents

[**Tasks Documentation** 3](#_Toc120179906)

[1. Retrospection - Feedback on sprint A 3](#_Toc120179907)

[DoD check for this task: 5](#_Toc120179908)

[2. Create basic Angular application 6](#_Toc120179909)

[File explanation: 6](#_Toc120179910)

[Test 7](#_Toc120179911)

[DoD check for this task: 7](#_Toc120179912)

[3. Add Angular Routing for Warehouse 8](#_Toc120179913)

[Test 8](#_Toc120179914)

[DoD check for this task: 9](#_Toc120179915)

[4. Add Angular page for Warehouse creation (with form and bulma styling) 10](#_Toc120179916)

[Test 12](#_Toc120179917)

[DoD check for this task: 12](#_Toc120179918)

[5. Add Angular page for Warehouse updating (with form and bulma styling) 12](#_Toc120179919)

[Test 14](#_Toc120179920)

[DoD check for this task: 14](#_Toc120179921)

[6. Add Angular page for Warehouse listing (with form and bulma styling) 15](#_Toc120179922)

[Test 16](#_Toc120179923)

[DoD check for this task: 16](#_Toc120179924)

[7. Add Angular service for Warehouse 17](#_Toc120179925)

[Test 17](#_Toc120179926)

[DoD check for this task: 18](#_Toc120179927)

[8. Add Angular Routing for Delivery 18](#_Toc120179928)

[9. Add Angular page for Delivery creation (with form and bulma styling) 18](#_Toc120179929)

[Test 20](#_Toc120179930)

[DoD check for this task: 20](#_Toc120179931)

[10. Add Angular page for Delivery updating (with form and bulma styling) 21](#_Toc120179932)

[Test 22](#_Toc120179933)

[DoD check for this task: 22](#_Toc120179934)

[11. Add Angular page for Delivery listing (with form and bulma styling) 22](#_Toc120179935)

[12. Add Angular service for Delivery 23](#_Toc120179936)

[13. Add Angular Routing for Trucks 23](#_Toc120179937)

[14. Add Angular page for Truck creation (with form and bulma styling) 23](#_Toc120179938)

[15. Add Angular page for Truck updating (with form and bulma styling) 23](#_Toc120179939)

[16. Add Angular page for Truck listing (with form and bulma styling) 23](#_Toc120179940)

[17. Add Angular service for Truck 23](#_Toc120179941)

[18. Add Angular Routing for Warehouse routes 24](#_Toc120179942)

[19. Add Angular page for warehouse route creation (with form and bulma styling) 24](#_Toc120179943)

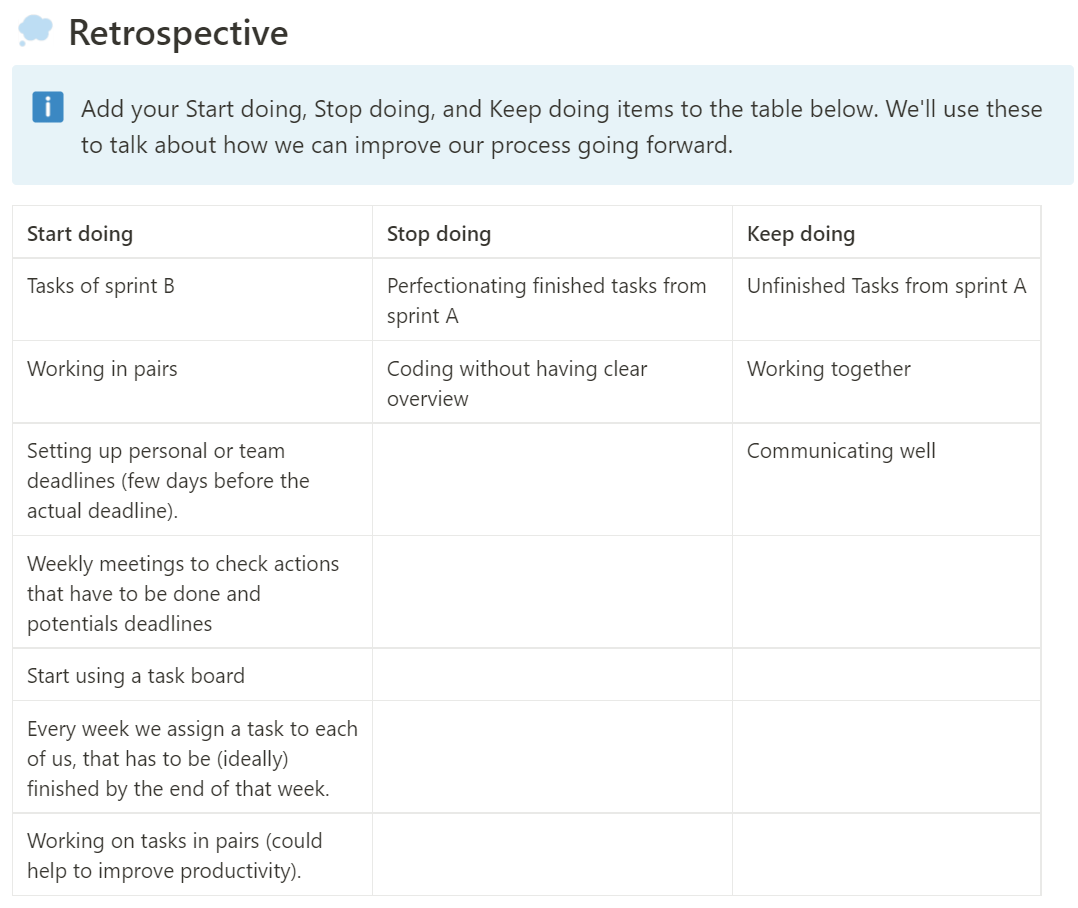
[20. Add Angular page for warehouse route updating (with form and bulma styling) 24](#_Toc120179944)

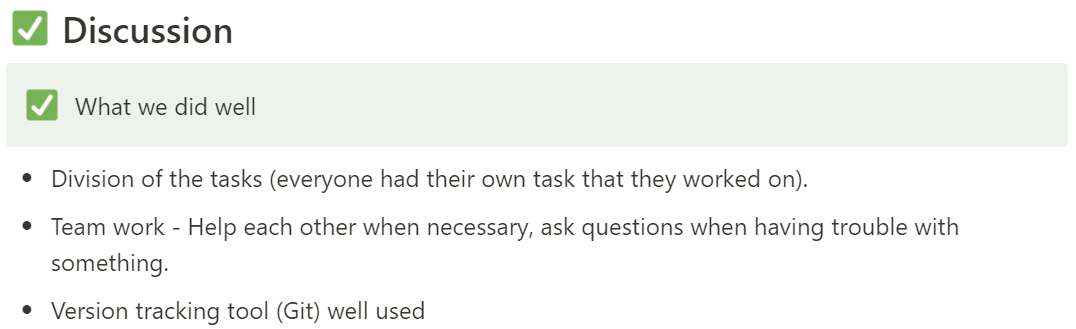
[21. Add Angular page for warehouse route listing (with form and bulma styling) 24](#_Toc120179945)

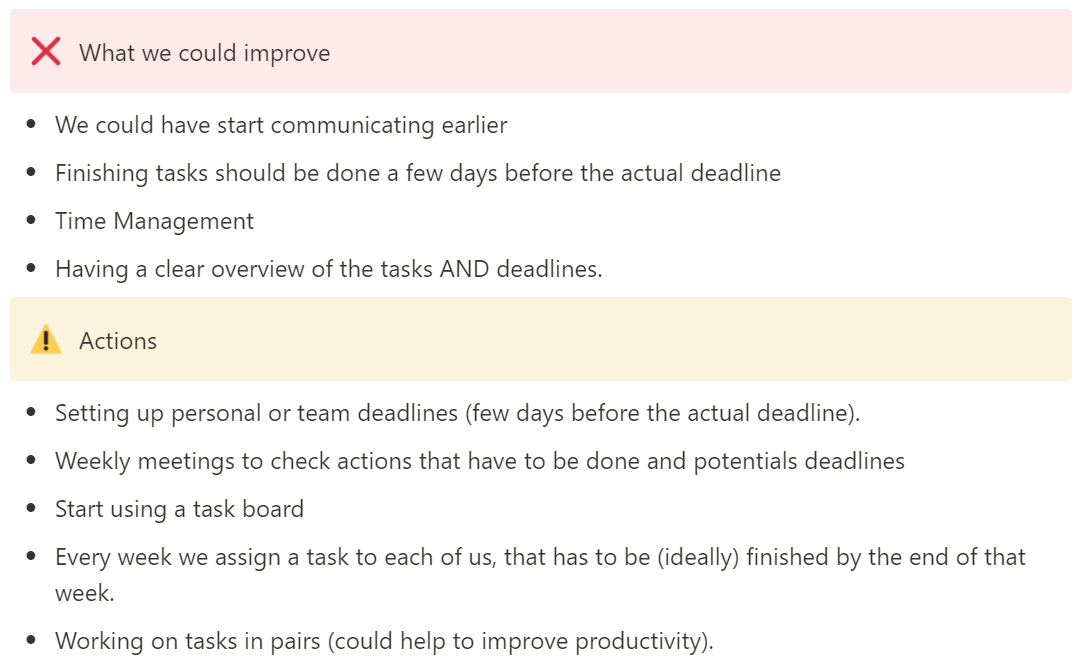
# **Tasks Documentation**

## 1. Retrospection - Feedback on sprint A

The retrospection was written and documented on notion. The topics (what went well, what could we improve, what should we do to do improve it?) that are associated with the sprint retrospection were discussed with the whole team, and the results of this meeting were assembled/collected together on a notion page, as shown below:







### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*The task was finished and the results of it were put on Notion*

*- [ ] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*Not applicable for this kind of task*

*- [ ] Test results documented*

*Not applicable for this kind of task*

*- [ ] (If applicable) Task builds without errors*

*Not applicable for this kind of task*

*- [ ] Task respected GIT version control (committed)*

*Not applicable for this kind of task*

*- [X ] General documentation for task is written*

*The retrospection documentation can be found before this checklist.*

*- [X] Task checked (and corrected if necessary) by group member*

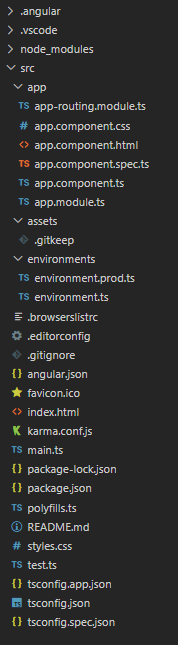
*Everyone was part of the meetings when we discussed everything and put it on notion, so everyone checked and agreed.*

* **Based on the checklist, we could conclude that this task was done.**

## 2. Create basic Angular application

For this task the angular CLI was installed with npm in the terminal.

The following command that is getting executed is *ng new file\_name*. This results in the following folder with the following default files:



The angular application can be run by executing the *ng serve --open* command. This will result in the application starting and immediately opening in the browser.

### File explanation:

Below we’ll discuss a few of the core files to get this angular application running:

**Index.html** – This is the base html file of the project. All the other html files used in the project will be imported here because it has <app-root> in its body.

**App.component.html –** This the app-root that is imported into base index.html file. All the components html files can be imported here in the same way as this was imported in the index.html file (by using < and *selectorName* and >).

**App.component.ts –** This file declares the selectorName (to import component into other files), and the file location where angular can find the templateUrl (html file) and styleUrls (css file).

**App.component.css –** This file manages the styling of the html file.

**Styles.css –** General styling file. This styling applies to the whole project.

**App.module.ts –** This file declares all the components files (declarations), imports necessary modules (for example FormsModule if you need to use NgModel, or the RoutingModule) and defines which component is the “bootstrap”, which is by default the AppComponent.

**App-routing-module.ts –** This file makes routing in angular possible. By defining routes users can go to a certain page/component by surfing to the url that is defined in the route path.

For this to work, in one of the html files (app.component.html), the <router-outlet> selector should be used/imported.

### Test

*Explanation:*

Because no code was written by ourselves for this, we tested this task by performing *ng serve --open*, with the expected result that the browser will open with a page that has the default text “hello world” (as we put it between two header tags in the app.component.html file).

*How to execute:*

1. Execute ng serve --open command in a terminal at the location of the angular project.
2. Check if “hello world” appears in the app that is opened in the browser
3. Check if the console doesn’t show any critical bugs/errors (browser dev tools).

*Result:*

**Need picture here**

We see “hello world” between two header tags as was expected, while no errors are being shown. So, the test is successful.

### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*The angular project was build and could run without problems and the results are as expected, comments in the files explain the purpose of certain features.*

*- [X] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*App could be run and when opening it in the browser we see the default text.*

*- [X] Test results documented*

*Can be found before the checklist*

*- [X] (If applicable) Task builds without errors*

*No errors are shown when performing the task.*

*- [X] Task respected GIT version control (committed)*

*Task was committed together with the truck frontend task (on 11 November 2022).*

*- [X ] General documentation for task is written*

*Can be found before the tests section.*

*- [X] Task checked (and corrected if necessary) by group member*

*Not yet*

* **Based on the checklist, we could conclude that this task was done.**

## 3. Add Angular Routing for Warehouse

When you go to app-routing-module.ts, you’ll see three routes defined. One for listing, one for creating, and one for updating.

A screenshot of a computer

Description automatically generated with medium confidence

Browsing to <http://localhost:4200/warehouses> will lead you to the Warehouses page that lists all the warehouses that are present in the database. In other words, this will load the WarehousesComponent that has this specific purpose and will be explained and shown in the “Add Angular page for Warehouse listing (with bulma styling)” task that can be found below.

Path defines the actual url path the users have to go to, while component is the actual Angular component that contains all the logic (html page and file logic).

### Test

*Explanation:*

User browses to the three defined routes (/warehouses, /warehouse/create, and /warehouse/W01) and expects to see the html files that are associated with the components that are assigned to each route.

*How to execute:*

1. Browse to /warehouses
2. Expect list with warehouses
3. Browse to /warehouse/create
4. Expect page with form to create a warehouse
5. Browse to /warehouse/W01
6. Expect page with Warehouse 1 information that you can update.

*Result:*

**Graphical user interface, application

Description automatically generated**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

All the URLs return the right and expected page, so we can say this test is successful.

### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*Routes were added to the routing module, comments were added after each path to explain what will be shown on that page.*

*- [X] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*Users can go to each URL to see the expected page.*

*- [X] Test results documented*

*Can be found before the checklist*

*- [X] (If applicable) Task builds without errors*

*No errors are shown when performing the task.*

*- [X] Task respected GIT version control (committed)*

*Task was committed to GitHub (on 11 November 2022).*

*- [X ] General documentation for task is written*

*Can be found before the tests section.*

*- [X] Task checked (and corrected if necessary) by group member*

*Not yet*

* **Based on the checklist, we could conclude that this task was done.**

## 4. Add Angular page for Warehouse creation (with form and bulma styling)

In src/app/components/warehouseadd you’ll find all the files are that are associated with this component.

Graphical user interface, text, email

Description automatically generated

The warehouseadd component typescript file contains all the logic including all the fields that are necessary for a warehouse and the logic to actually save a warehouse (which will only be done if all the fields have data and are not undefined).

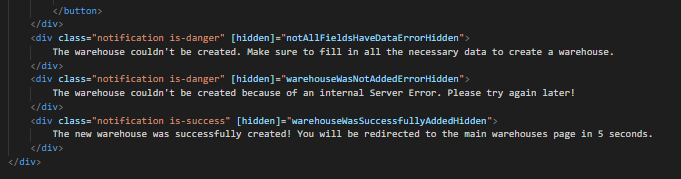
Text

Description automatically generated

Graphical user interface, text, application

Description automatically generatedThe template file is using bulma again for styling, and has a form with fields. ngModel is used to immediately update the fields data, so that when pressing the button to create a warehouse, the right data will be used to create that warehouse.

Further we also have a section that is associated with the variables in the logic file that end with “hidden”. This will be shown when the value of these variables is false.



Styling is the same as used in the other files

A screenshot of a computer

Description automatically generated with medium confidence

### Test

*Explanation:*

***UNIT TEST STILL HAS TO BE DONE***

*How to execute:*

**PERFORM UNIT TEST**

### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*All the logic to create warehouses is written and commented, the html page to show the form with all the fields and be directly connected with the logic page thanks to ngModel is also fully functional.*

*When there are errors or a warehouse is successfully created, the right message will be shown on the page.*

*- [X] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*Not yet*

*- [X] Test results documented*

*Not yet*

*- [X] (If applicable) Task builds without errors*

*No errors are shown when performing the task.*

*- [X] Task respected GIT version control (committed)*

*Task was committed to GitHub (on 11 November 2022).*

*- [X ] General documentation for task is written*

*Can be found before the tests section.*

*- [X] Task checked (and corrected if necessary) by group member*

*Not yet*

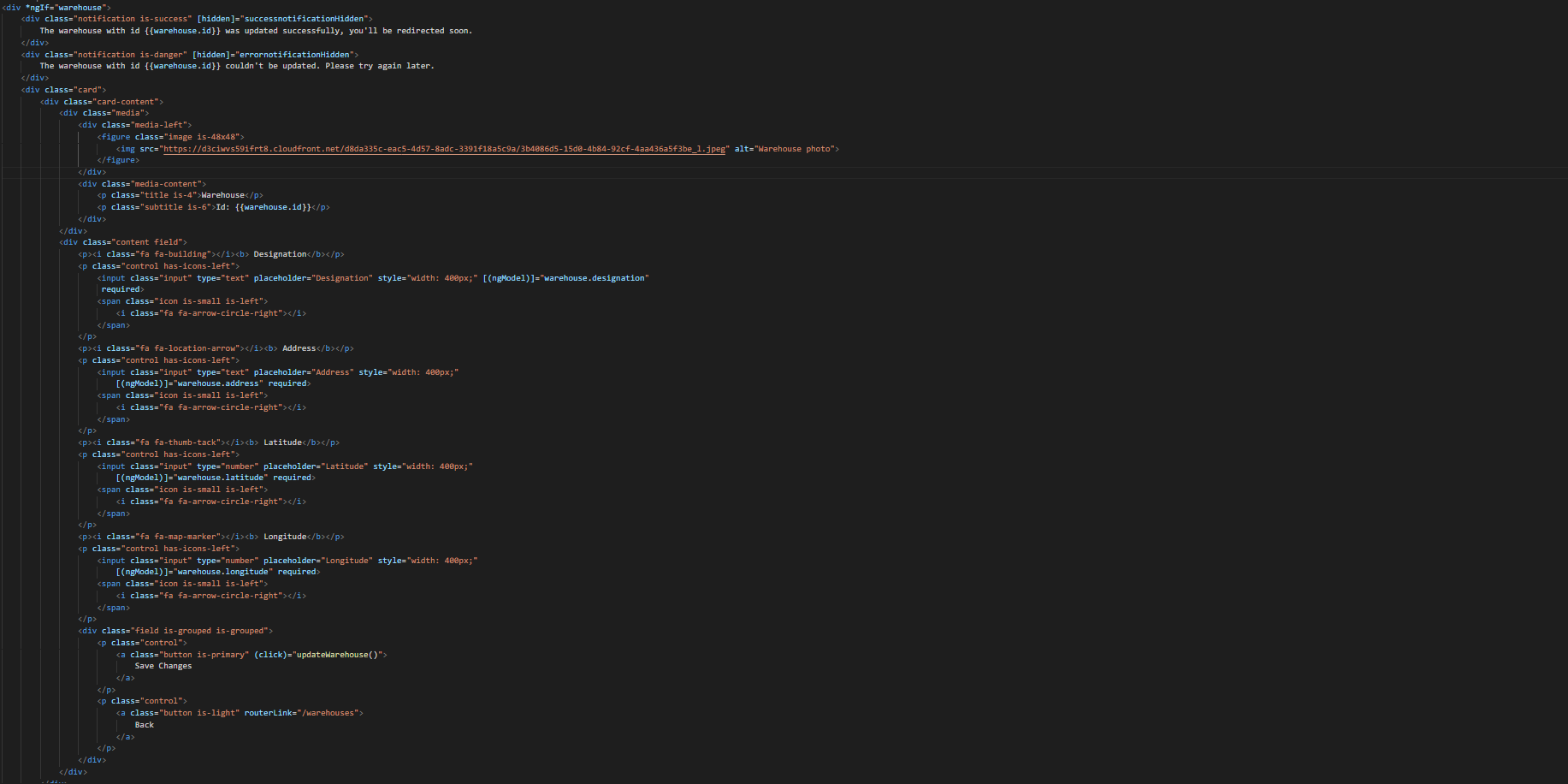
## 5. Add Angular page for Warehouse updating (with form and bulma styling)

In src/app/components/warehouse-detail you’ll find all the files are that are associated with this component.

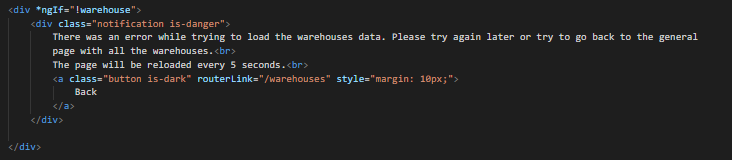
Text

Description automatically generated

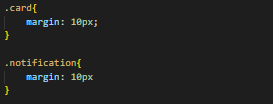
The warehouse detail component typescript file contains all the logic to load the specific warehouse data based on the entered Id in the url. It also contains the logic to update the warehouse data by using the warehouse service.

Only if the warehouse variable contains data (so id exists and warehouse is successfully loaded), a form will be shown with the warehouse data. When a user changes one of the fields, this will immediately affect the data in the warehouse variable because of ngModel. When pressing the save changes button, the update method will be called which updates the warehouse.

Further we also have a section that shows an error when no warehouse could be found.



For styling the following was used to create some space next to the elements:



### Test

*Explanation:*

***UNIT TEST STILL HAS TO BE DONE***

*How to execute:*

**PERFORM UNIT TEST**

### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*All the logic to import the specific warehouse and update a warehouse is written and commented, the html page to show the form with all the fields and be directly connected with the logic page thanks to ngModel is also fully functional (thanks to this the fields will be filled in with the warehouse data, and when something changes the warehouse data in typescript logic file will also be updated with the new value). When no data is available, the page will show an error.*

*- [X] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*Not yet*

*- [X] Test results documented*

*Not yet*

*- [X] (If applicable) Task builds without errors*

*No errors are shown when performing the task.*

*- [X] Task respected GIT version control (committed)*

*Task was committed to GitHub (on 11 November 2022).*

*- [X ] General documentation for task is written*

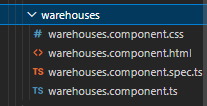
*Can be found before the tests section.*

*- [X] Task checked (and corrected if necessary) by group member*

*Not yet*

## 6. Add Angular page for Warehouse listing (with form and bulma styling)

In src/app/components/warehouses you’ll find all the files are that are associated with this component.



The warehouse component typescript file contains all the logic to load the warehouse data. Text

Description automatically generated

The template/html file will use this loaded data to show the warehouses in a card styled by the bulma styling framework. To import the data, {{ }} is used. \*ngFor is used to loop through the warehouses list so that we can show every and each warehouse that is in it.

Also a button for updating or creating a warehouse is present. Thanks to routerLink, users will be redirected to the associated route with that button.

Text

Description automatically generated

### Test

*Explanation:*

***UNIT TEST STILL HAS TO BE DONE***

*How to execute:*

**PERFORM UNIT TEST**

### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*All the logic to import warehouses is written and commented, the html page to show those data in a bulma card is also coded and fully functional. When no data is available, the page will show an error.*

*- [X] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*Not yet*

*- [X] Test results documented*

*Not yet*

*- [X] (If applicable) Task builds without errors*

*No errors are shown when performing the task.*

*- [X] Task respected GIT version control (committed)*

*Task was committed to GitHub (on 11 November 2022).*

*- [X ] General documentation for task is written*

*Can be found before the tests section.*

*- [X] Task checked (and corrected if necessary) by group member*

*Not yet*

## 7. Add Angular service for Warehouse

The service file can be found back in the src/app/services folder. This warehouse.service.ts file contains all the data to make a connection with the warehouse backend API.

Text

Description automatically generated

By importing HttpClient we can execute GET, POST, PUT calls to an external API. This is done to retrieve and update the warehouse data and create a new warehouse. All calls we be sent with the content type application/json so that we can sent and retrieve json data. All methods return an observable so that other functions that use those methods of the service can subscribe to it.

Text

Description automatically generated

### Test

*Explanation:*

***UNIT TEST STILL HAS TO BE DONE***

*How to execute:*

**PERFORM UNIT TEST**

### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*All the logic to execute GET, PUT, and POST call to the warehouse API backend is written and fully functional (as long as the backend is working).*

*- [X] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*Not yet*

*- [X] Test results documented*

*Not yet*

*- [X] (If applicable) Task builds without errors*

*No errors are shown when performing the task.*

*- [X] Task respected GIT version control (committed)*

*Task was committed to GitHub (on 11 November 2022).*

*- [X ] General documentation for task is written*

*Can be found before the tests section.*

*- [X] Task checked (and corrected if necessary) by group member*

*Not yet*

## 8. Add Angular Routing for Delivery

Please take a look at [*3. Add Angular Routing for Warehouse*](#_3._Add_Angular), as the Routing for Delivery works the same way (and is made and tested the same way) as this task.

## 9. Add Angular page for Delivery creation (with form and bulma styling)

Quite similar to [*4. Add Angular page for Warehouse creation (with form and bulma styling)*](#_4._Add_Angular)*.*

In src/app/components/deliveryadd you’ll find all the files are that are associated with this component.

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generatedText

Description automatically generatedThe deliveryadd component typescript file contains all the logic including all the fields that are necessary for a delivery and the logic to actually save a delivery (which will only be done if all the fields have data and are not undefined). Additionally, it will also load warehouse ids that users can select from a dropdown menu, this way they can only select existing warehouses for their delivery.

The html page is build in the same way as the warehouses one ([*4. Add Angular page for Warehouse creation (with form and bulma styling)*](#_4._Add_Angular)), only with different fields. The only difference is that here a dropdown menu is used with Ids based on the warehouses data that exist on the db, this way we can only make deliveries with valid warehouse Ids. For this, \*ngFor is used so that we have an option for each warehouse (id) in the warehouseIds list. To have this functional, both the value as the actual text of option have to be that id. We also have another kind of field type (date).

*The code of this can be found below*

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

### Test

*Explanation:*

***UNIT TEST STILL HAS TO BE DONE***

*How to execute:*

**PERFORM UNIT TEST**

### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*All the logic to load the warehouseIds, save a new delivery, have a dropdown for the warehouseIds, protection when no warehouseIds are present and a new delivery shouldn’t be able to be created (incl. messages) is written and commented. Error message will be shown when no id was loaded because of server problems or no warehouses existing in the db. So the whole part is fully functional.*

*- [X] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*Not yet*

*- [X] Test results documented*

*Not yet*

*- [X] (If applicable) Task builds without errors*

*No errors are shown when performing the task.*

*- [X] Task respected GIT version control (committed)*

*Task was committed to GitHub (on 11 November 2022).*

*- [X ] General documentation for task is written*

*Can be found before the tests section.*

*- [X] Task checked (and corrected if necessary) by group member*

*Not yet*

## 10. Add Angular page for Delivery updating (with form and bulma styling)

In src/app/components/delivery-detail you’ll find all the files that are associated with this component.

Graphical user interface, text

Description automatically generated

Text

Description automatically generatedThe delivery detail component typescript file contains all the logic to load the specific delivery data based on the entered Id in the URL. It also contains the logic to load warehouses and push all their Ids to a list, so that users can only update the warehouseId with an existing warehouseId. If this function does not work, an error message will be shown by setting hidden value of min1WarehouseSuccessfullyLoaded value to false.

Text

Description automatically generatedIt also contains the logic to update the delivery data by using the delivery service. This first checks if all the fields have data, and will then use the same warehouse variable (from which the data could have been updated by the user and by using ngModel) to update that specific warehouse in the db. If successful a success message will be shown by setting hidden value to false, if an error occurs an error message will be shown by setting that hidden value to false.

The template page is the same as can be seen on the creation page ([9. Add Angular page for Delivery creation (with form and bulma styling)](#_9._Add_Angular)), with the little difference that on the updating page already data of a delivery with a specific id is filled in.

### Test

*Explanation:*

***UNIT TEST STILL HAS TO BE DONE***

*How to execute:*

**PERFORM UNIT TEST**

### DoD check for this task:

*- [X ] Task is delivered (code is written, completely functional and comments explain the parts)*

*All the logic to load the warehouseIds, retrieve and update a specific delivery, have a dropdown for the warehouseIds, protection when no warehouseIds are present and a delivery shouldn’t be able to be updated (incl. messages) is written and commented. Error message will be shown when no id was loaded because of server problems or no warehouses existing in the db. So the whole part is fully functional.*

*- [X] Task is tested (if applicable) → Unit test for code | Personal test for other kind of tasks*

*Not yet*

*- [X] Test results documented*

*Not yet*

*- [X] (If applicable) Task builds without errors*

*No errors are shown when performing the task.*

*- [X] Task respected GIT version control (committed)*

*Task was committed to GitHub (on 11 November 2022).*

*- [X ] General documentation for task is written*

*Can be found before the tests section.*

*- [X] Task checked (and corrected if necessary) by group member*

*Not yet*

## 11. Add Angular page for Delivery listing (with form and bulma styling)

Please take a look at [*6. Add Angular page for Warehouse listing (with form and bulma styling)*](#_6._Add_Angular)*,* as these files are made in the same way as the warehouses one, with the only difference that warehouses names are replaced by deliveries and other fields are used for the deliveries (id, deliveryDate, timeToPickup, timeToPlace, warehouseID, and weight).

## 12. Add Angular service for Delivery

This is the same as [*7. Add Angular service for Warehouse*](#_7._Add_Angular). The only difference is that a different URL is being used: https://localhost:5001/api/Deliveries

## 13. Add Angular Routing for Trucks

Please take a look at [*3. Add Angular Routing for Warehouse*](#_3._Add_Angular), as the Routing for Trucks works the same way (and is made and tested the same way) as this task. The only difference is that the route name is not Warehouse but Truck (so three routes: /trucks, /truck/create, and /truck/:id).

## 14. Add Angular page for Truck creation (with form and bulma styling)

Please take a look at [*4. Add Angular page for Warehouse creation (with form and bulma styling)*](#_4._Add_Angular)*,* as these files are made in the same way as the warehouses one, with the only difference that warehouses names are replaced by trucks and other fields are used for the trucks (tare, load\_capacity, max\_battery\_charge, autonomy, and fast\_charging\_time).

## 15. Add Angular page for Truck updating (with form and bulma styling)

Please take a look at [*5. Add Angular page for Warehouse updating (with form and bulma styling)*](#_5._Add_Angular)*,* as these files are made in the same way as the warehouses one, with the only difference that warehouses names are replaced by trucks and other fields are used for the trucks (tare, load\_capacity, max\_battery\_charge, autonomy, and fast\_charging\_time).

## 16. Add Angular page for Truck listing (with form and bulma styling)

Please take a look at [*6. Add Angular page for Warehouse listing (with form and bulma styling)*](#_6._Add_Angular)*,* as these files are made in the same way as the warehouses one, with the only difference that warehouses names are replaced by trucks and other fields are used for the trucks (tare, load\_capacity, max\_battery\_charge, autonomy, and fast\_charging\_time).

## 17. Add Angular service for Truck

This is the same as [*7. Add Angular service for Warehouse*](#_7._Add_Angular). The only difference is that a different URL is being used: http://localhost:3000/api/v1/trucks.

## 18. Add Angular Routing for Warehouse routes

Please take a look at [*3. Add Angular Routing for Warehouse*](#_3._Add_Angular), as the Routing for Warehouse routes works the same way (and is made and tested the same way) as this task. The only difference is that the route name is not Warehouse but Truck (so three routes: /warehouse-routes, /warehouse-route/create, and /warehouse-route/:id).

## 19. Add Angular page for warehouse route creation (with form and bulma styling)

Please take a look at [*9. Add Angular page for Delivery creation (with form and bulma styling)*](#_9._Add_Angular)*,* as these files are made in the same way as the delivery one, with the only difference that delivery names are replaced by warehouse routes and other fields are used for the warehouse routes (departure\_warehouseId, destination\_warehouseId, distance, time, used\_battery, and extra\_time\_when\_charging\_required). The departure and destination warehouseId is also made possible by using \*ngFor for the options, as is the same case for the warehouseId field of the delivery. So, for this the page also has to load the warehouse Ids successfully or it won’t work and be possible to create a new route.

## 20. Add Angular page for warehouse route updating (with form and bulma styling)

Please take a look at [*10. Add Angular page for Delivery updating (with form and bulma styling)*](#_10._Add_Angular)*,* as these files are made in the same way as the delivery one, with the only difference that delivery names are replaced by warehouse routes and other fields are used for the warehouse routes (departure\_warehouseId, destination\_warehouseId, distance, time, used\_battery, and extra\_time\_when\_charging\_required). The departure and destination warehouseId dropdown is also made possible by using \*ngFor for the options, as is the same case for the warehouseId field of the delivery. Users can only select existing warehouseIds from the list and not enter it manually.

So, for this creation page to work successfully, the page also has to load the warehouse Ids successfully or it won’t work and be possible to create a new route.

## 21. Add Angular page for warehouse route listing (with form and bulma styling)

Please take a look at [*6. Add Angular page for Warehouse listing (with form and bulma styling)*](#_6._Add_Angular)*,* as these files are made in the same way as the Warehouses one, with the only difference that Warehouse names are replaced by warehouse routes and other fields are used for the warehouse routes (departure\_warehouseId, destination\_warehouseId, distance, time, used\_battery, and extra\_time\_when\_charging\_required).