ADIN

(ADAPTIVE INTERFACES)

Make adaptability your main power

User Manual

SHOP4CF

Contents

Introduction

The porpuse of this document is explain the user what is tof the application, how to set it up and show how to use its main features.

ADIN - Architecture

ADIN is a web-based application capable of assisting employees in the manufacturing sector during their working time. ADIN helps avoid human error and reduce stress by providing the most relevant and specific information needed by the user, for example by guiding workers to perform tasks.

The application is comprised of four main blocks:

- Frontend (FE): Generates the interface with which the user interacts;
- Backend API (BE-API): It is in charge of defining the endpoints that provide data to the FE or receive information from it. Additionally, it defines the data models of the entities used by the application and shape the database.
- Backend Notification Handler (BE-NH): It handles the bidirectiona communication between ADIN and with other components via FIWARE.
- Database (PostgreSQL): It is where is stored all the necessary data that the application needs for its correct operation.

Aitor Toichoa Eyam
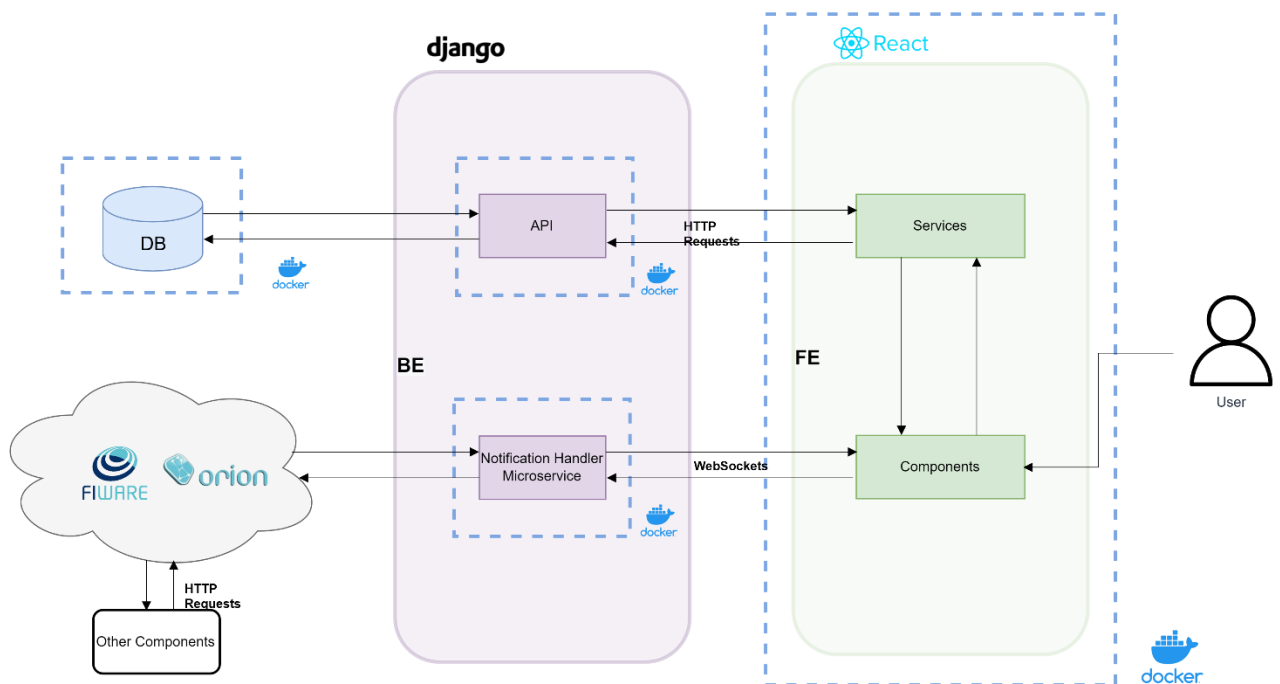
Component Access

ADIN component can be found at SHOP4CF repository of RAMP Docker Registry.

The component is divided into several artifacts. In order to use ADIN, it is necessary to download the following artifacts:

- *adin-api*
- *adin-fe*
- *adin-nh*
- *adin-db*

Software requirements

To use ADIN component, it is necessary to have Docker installed and a device able to run a web application. Additionally, it is advisible to run the application in a browser such as Google Chrome or similar.

Set up

Once docker is installed, it is required to download the images of the artifacts, *adin-api, adin-fe, adin-nh (Note: download the latest version of them)* from RAMP. In the next section is explained how to run the application using docker compose.

Docker Compose approach

Docker-compose requires having download all the images of the artifacts. Create a file named *docker-compose,*write in the document the following commands and and save it with the extension *.yml* (this can be done in a notepad program as NotePad++):

```
version: "3"

services:

    adin-api:

        image: docker.ramp.eu/shop4cf/adin-api:5.0.0

        container_name: adin-api

        command: bash -c "python manage.py runserver 0.0.0.0:8000"

        environment:

            - POSTGRES_USER= YOUR_USER

            - POSTGRES_PASSWORD=YOUR_PASSWORD

            - POSTGRES_DB=YOUR_DB

            - DB_HOST=adin-db

            - DB_PORT= 5432
```

Aitor Toichoa Eyam

```yaml
    ports:

        - "8000:8000"

    depends_on:

        - adin-db


adin-db:

    image: postgres:14

    environment:

        - POSTGRES_USER=YOUR_USER

        - POSTGRES_PASSWORD=YOUR_PASSWORD

        - POSTGRES_DB=local_adin_db_2

    volumes:

        - pgdata:/var/lib/postgresql/data

        - ./YOUR_DIRECTORY_TO_DATABASE_BACKUP:/docker-entrypoint-
initdb.d/init.sql:ro

    ports:

        - "5432:5432"


adin-fe:

    image: docker.ramp.eu/shop4cf/adin-fe:1.4.1

    container_name: adin-fe

    command: npm start

    depends_on:

      - adin-api

      - adin-nh

    ports:

      - "3000:3000"

    expose:

      - "3000"


adin-nh:

    image: docker.ramp.eu/shop4cf/adin-nh:1.3.0

    container_name: adin-nh

    command: bash -c "python bridge.py runserver"
```

Aitor Toichoa Eyam

```
        depends_on:
            - orion
        ports:
            - "54100:54100"
            - "8080:8080"
        expose:
            - "54100"
            - "8080"


    mongo:
        image: mongo:3.2
        command: --nojournal
        container_name: mongo


    orion:
        image: fiware/orion-ld:latest
        links:
            - mongo
        ports:
            - "1026:1026"
        command: -dbhost mongo -logLevel DEBUG -logForHumans
        container_name: orion
        expose:
            - "1026"
        depends_on:
            - mongo


volumes:
    pgdata:
```

First of all, to run the application, docker should be initialized. Once docker is running, navigate in the shell (e.g.: cmd) to the path where *docker-compose.yml* is located and execute the command: **_docker-compose up_**.

*Note: The command **docker-compose up -d** will execute the app in the background (detached).*

Aitor Toichoa Eyam

To check if the application has been deployed correctly, check at Docker Deskopt the *Containers/App* section:
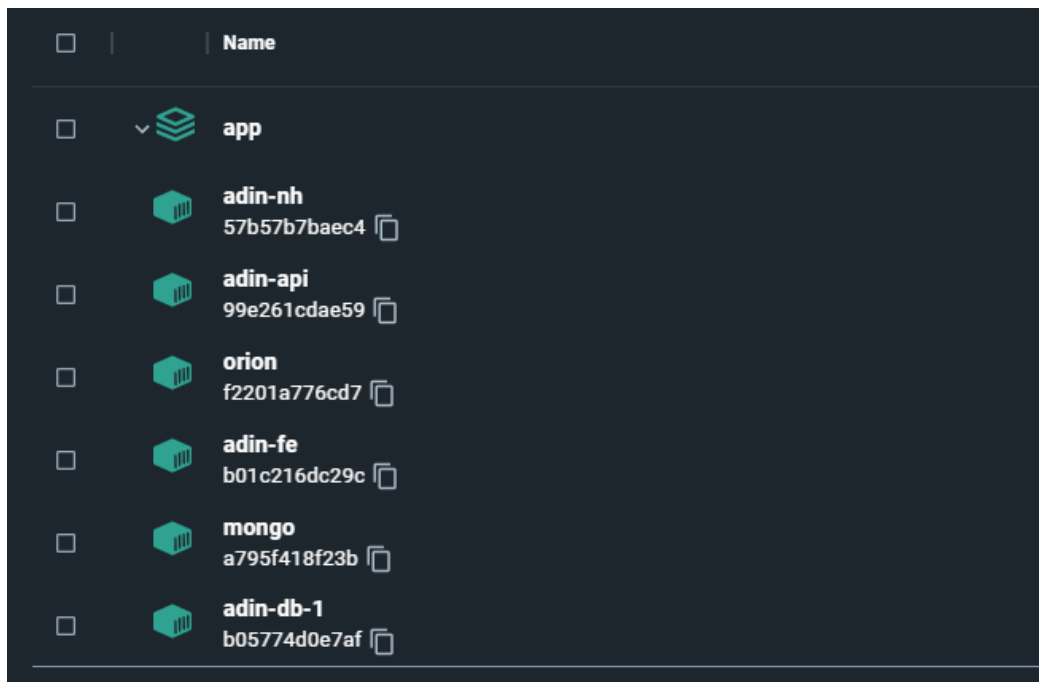
To stop the application, use the command: ***docker-compose down***.

Application Sections

The application is divided into two main sections: *Home Page* and *Dashboard*.

Home Page

This is the first page of the application where a user can perform to main actions:

- Log in: If the user has already an account, can log in and be redirected to the dashboard.
- Sign up: If the user doesn't have an account, can fill in the necessary data to register into the application.

Aitor Toichoa Eyam

Log in

To log in to the application the user should press the *log in* button which is located in the upper right section of the window.
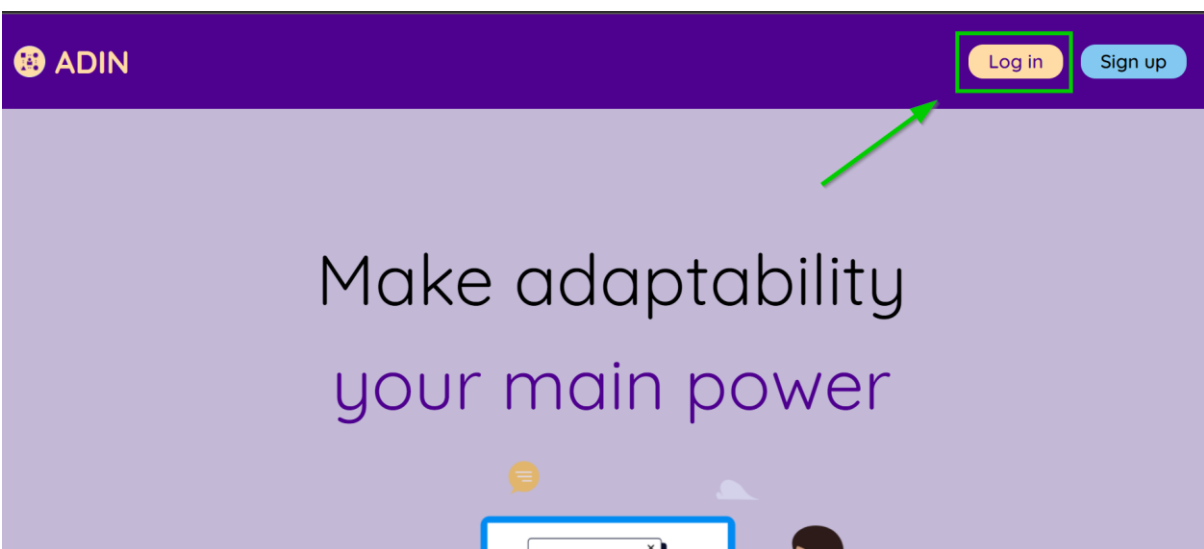
After pressing the button, it will appear a popup window in which the user should fill two fields, username and password.

Aitor Toichoa Eyam
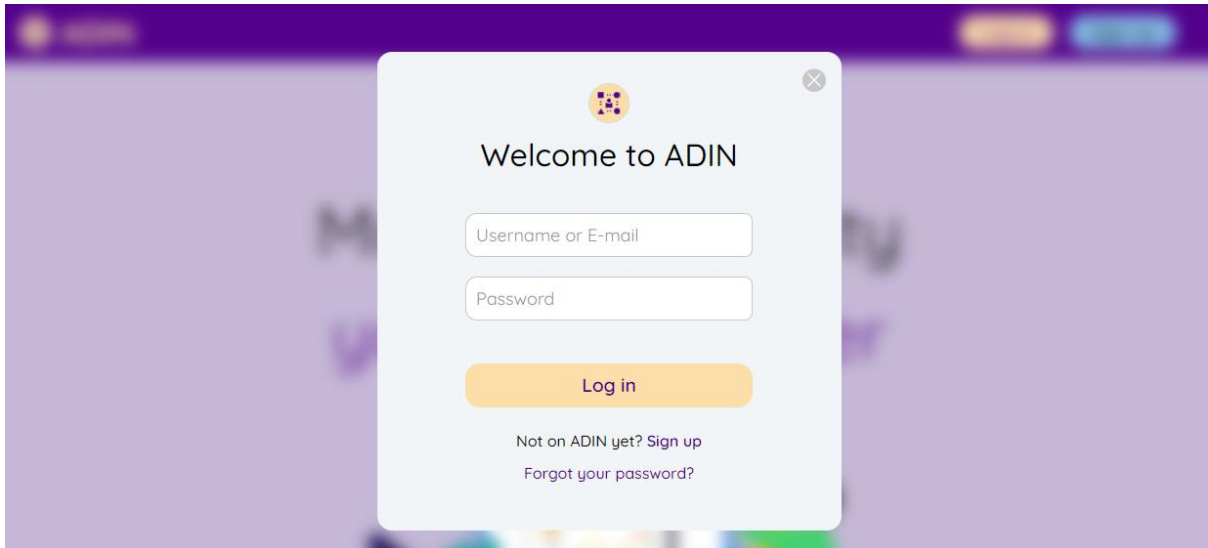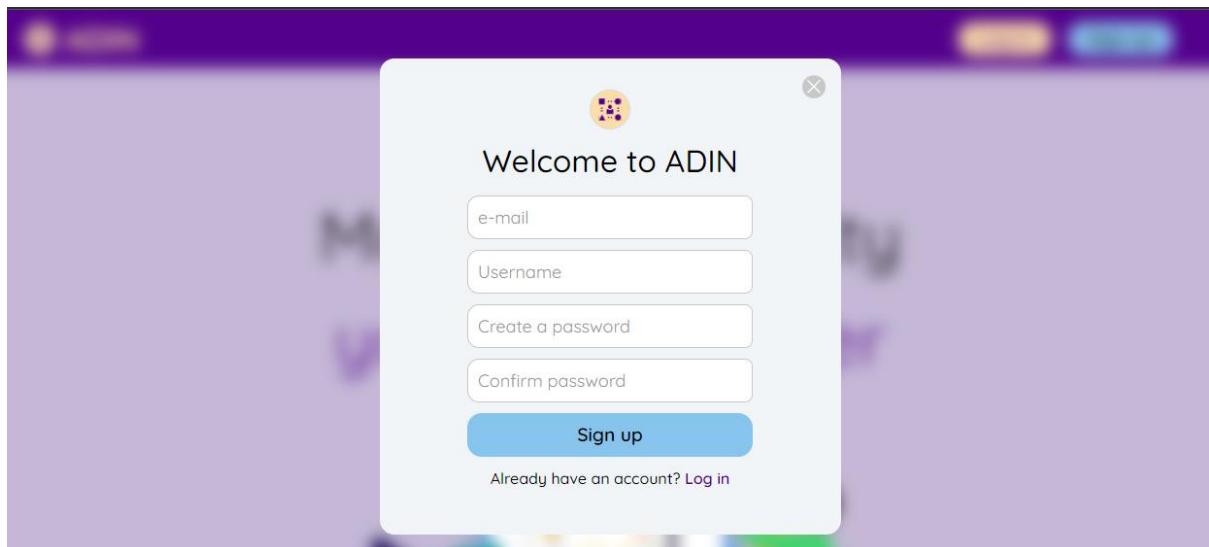
Once the username/e-mail and password have been introduced, by pressing the *Log in* button of the popup window, the user will be redirected to the Dashboard.
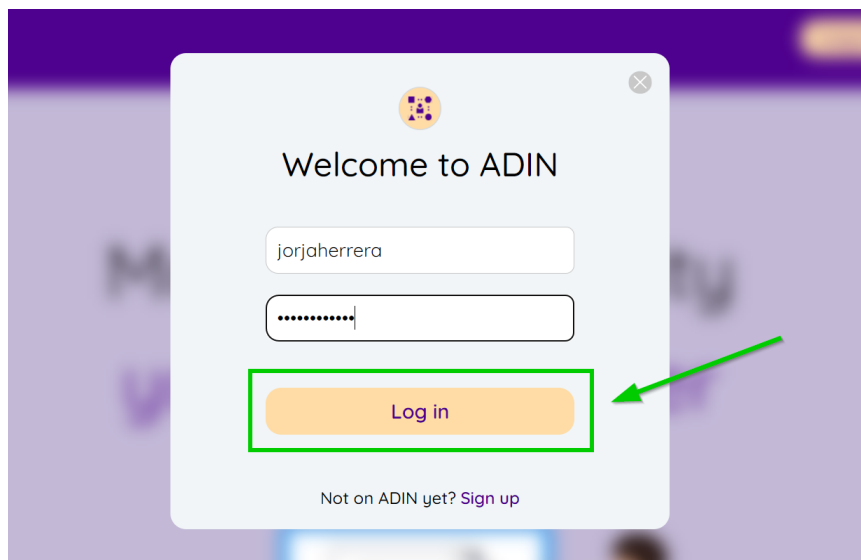
<u>Sign up</u>

Similar to the login process, in order to sign up in the application, the user should press the *sign up* button located in the top right corner of the screen. A popup window will emerge asking the user to introduce data to generate a profile.

Aitor Toichoa Eyam

Once the fields are filled, the user should select the general screen configuration that prefers.
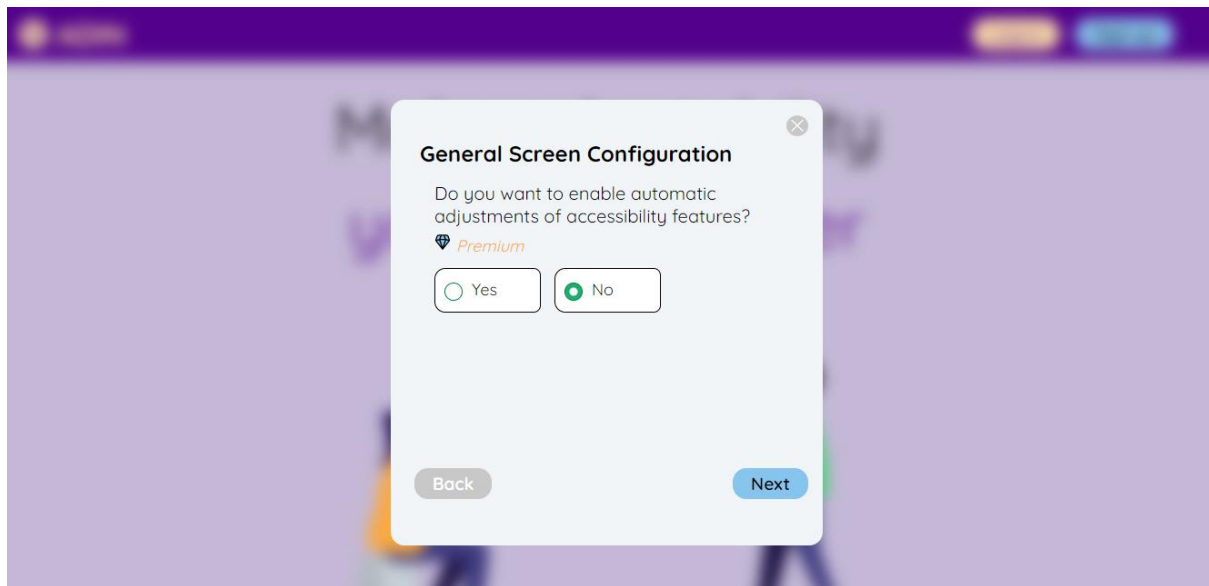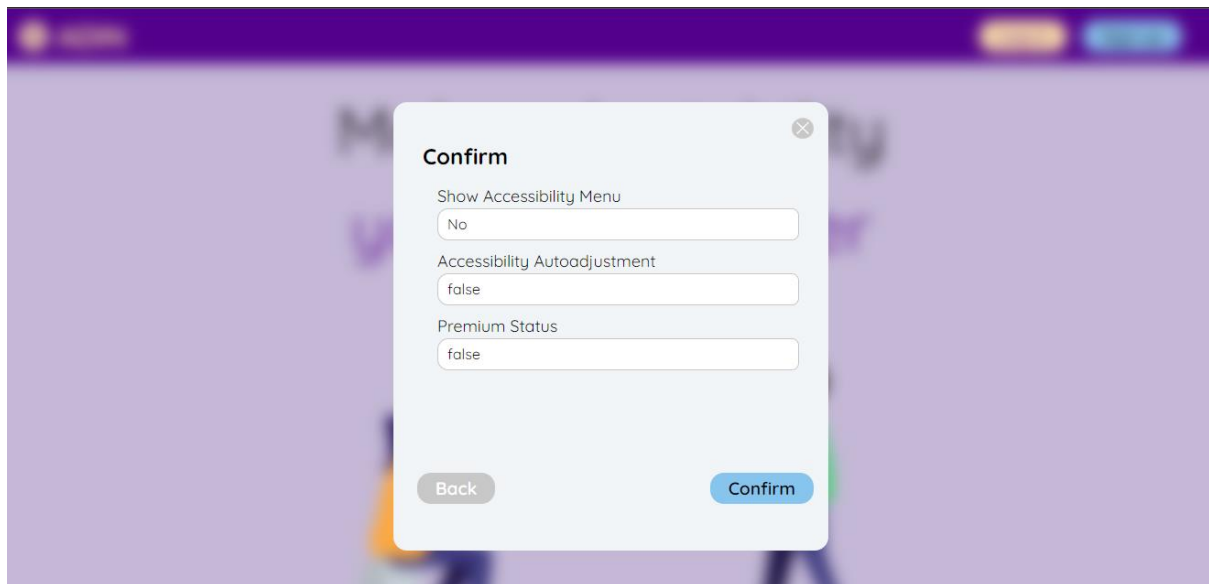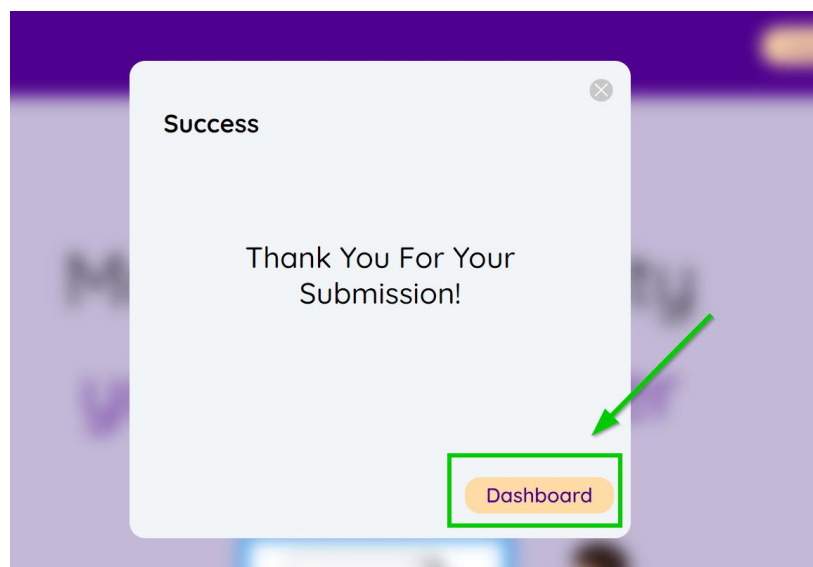
Then the use has can confirm the introduced data. If everything is correct, by pressing the *confirm* button the user will see a message confirming that the registration has been succeeded and a Dashboard button.
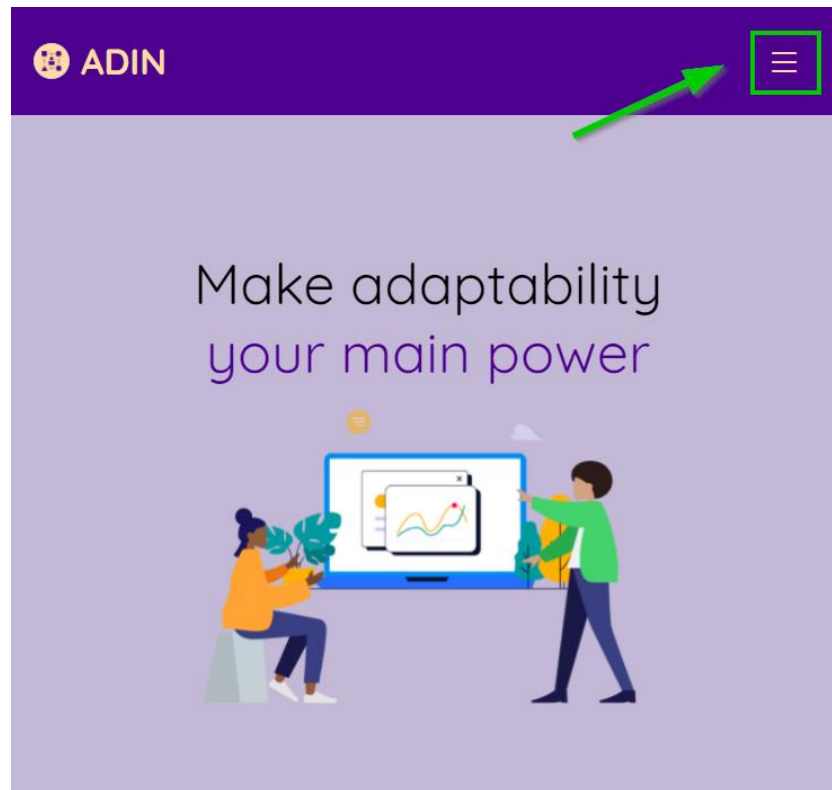

Aitor Toichoa Eyam
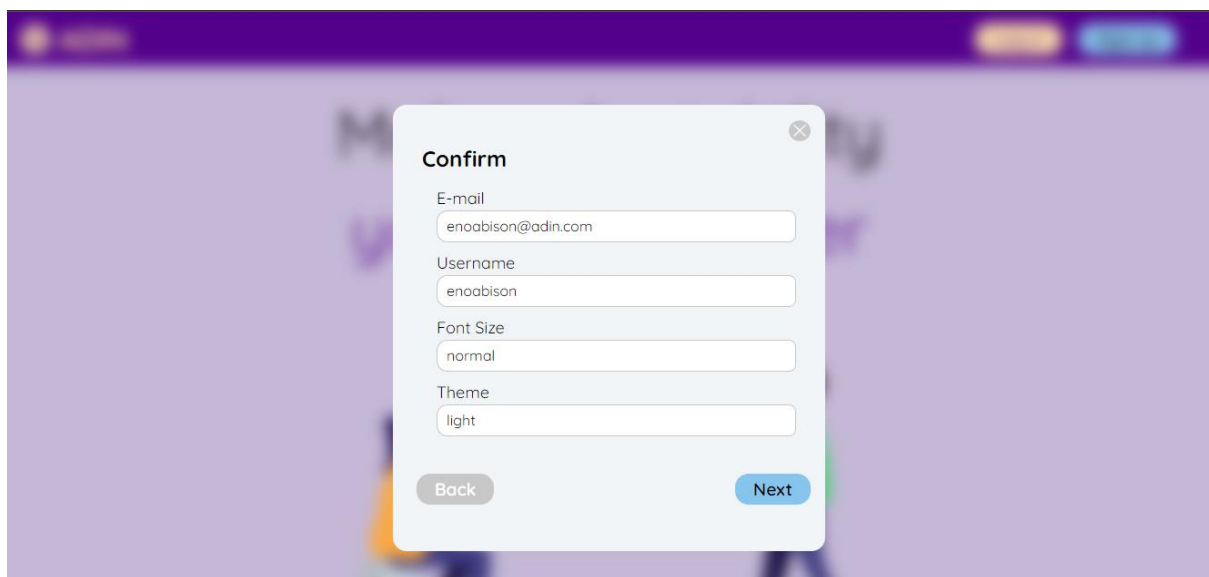
Pressing the *Dashboard* button will lead the user to the Dashboard.

Small screen

If the screen where the application is being displayed is reduced to a certain size, the items displayed on the navigation bar situated on top of the screen will disappear, being displayed in its place a hamburger menu.
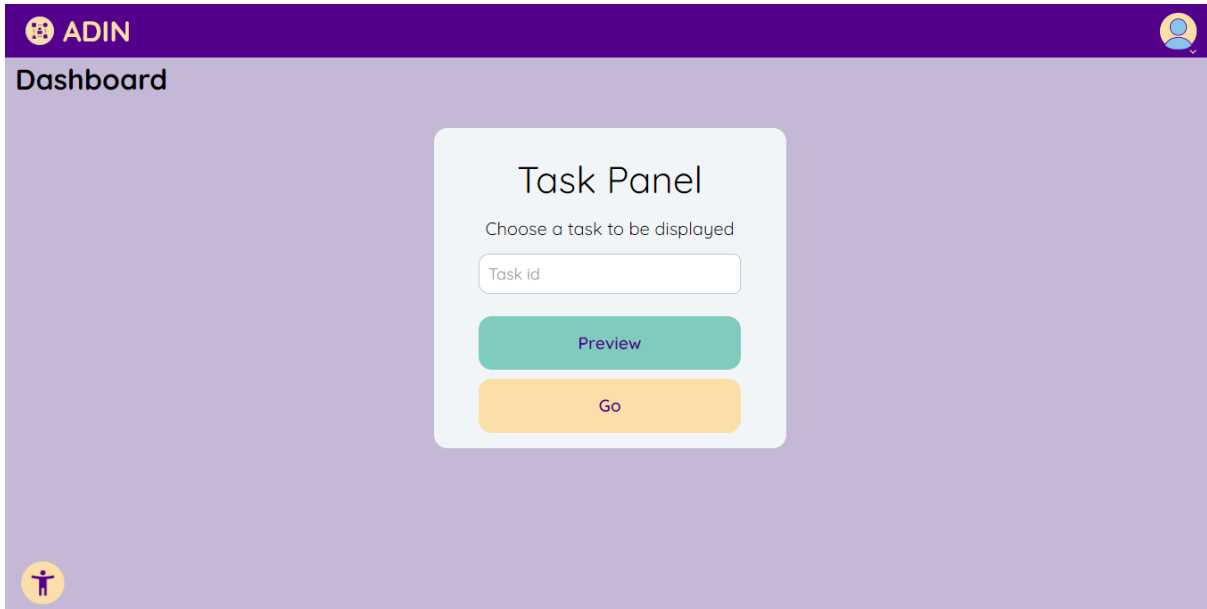
Aitor Toichoa Eyam

By clicking the hamburger icon, a menu will be displayed on the screen with the same buttons that were contained in the navigation bar. The user can make the same procedures of login and sign up described before.
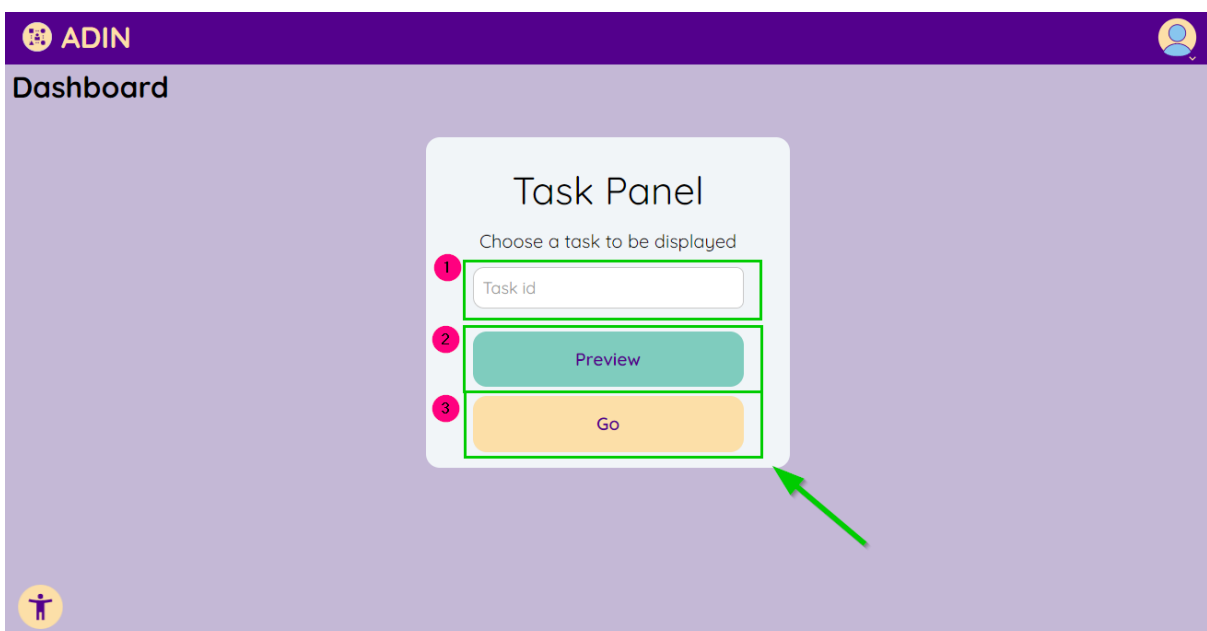

Aitor Toichoa Eyam
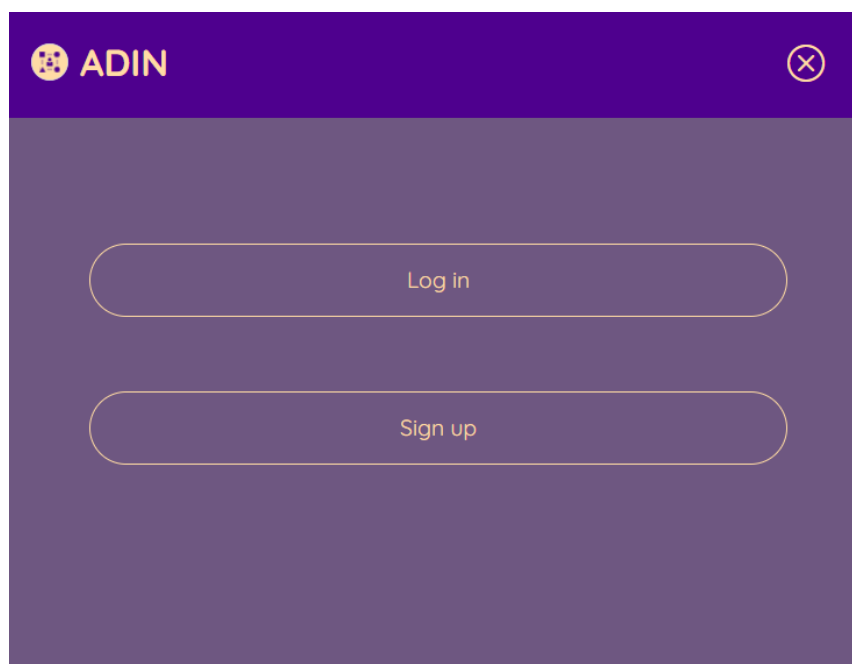
Dashboard

On this page, it is represented the *Task Panel*. The *Task Panel* is meant to assist the user through the development of a task by showing the required steps and information to fulfil without any risk the selected task.

The first element that the user will see is a panel to choose the id of the task to complete.

By introducing the id number of the task and pressing the *Go* button the application shows the *Task Panel* with the information of the selected task. Additionally, the user can preview the task that has selected by pressing the *Preview* button.

Aitor Toichoa Eyam

ADIN

Dashboard

Preview Task

Task ID: 1

Description: In this task the user will have to assemble a wooden box by fasten all the bolts with the assistance of a cobot. Additionally, the user will wear an EEG headset in order to share their emotional data with the cobot.

Number of Steps: 7

List of Components: Bolts

List of Tools: Screwdriver

List of Safety Tools: Gloves, Helmet

Next

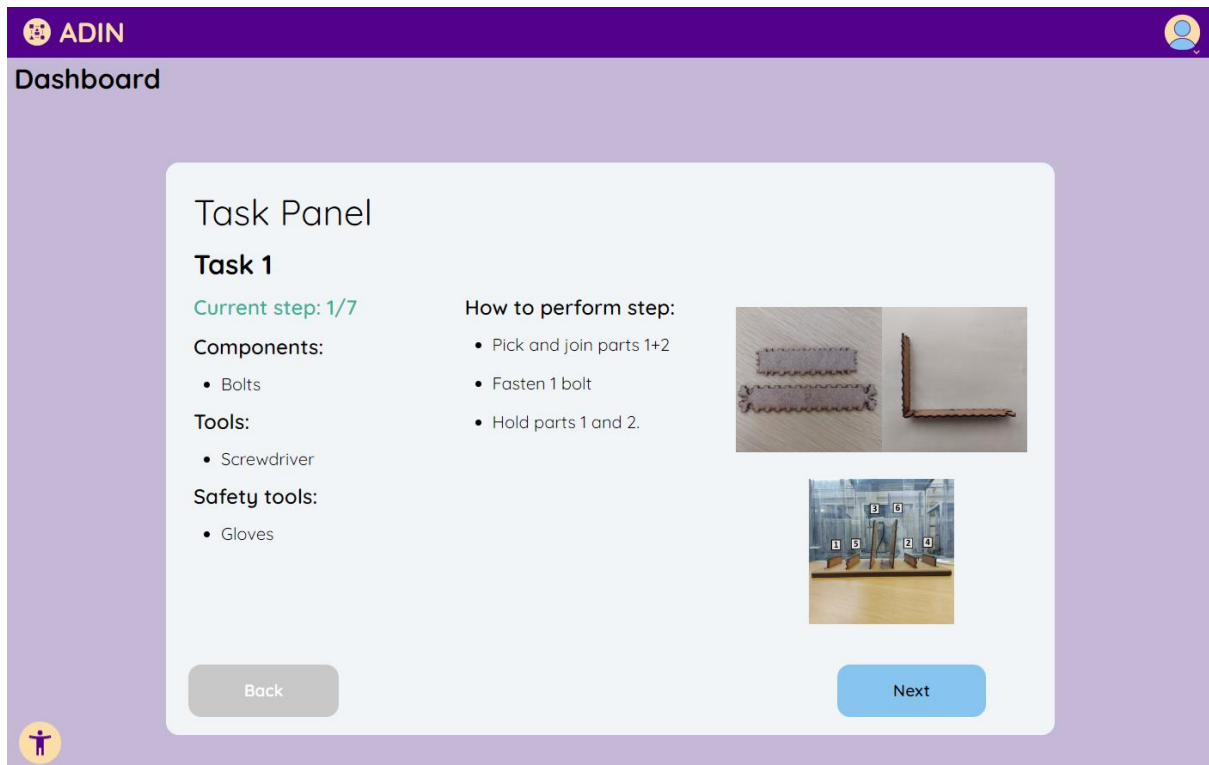ADIN

Dashboard

Preview Task
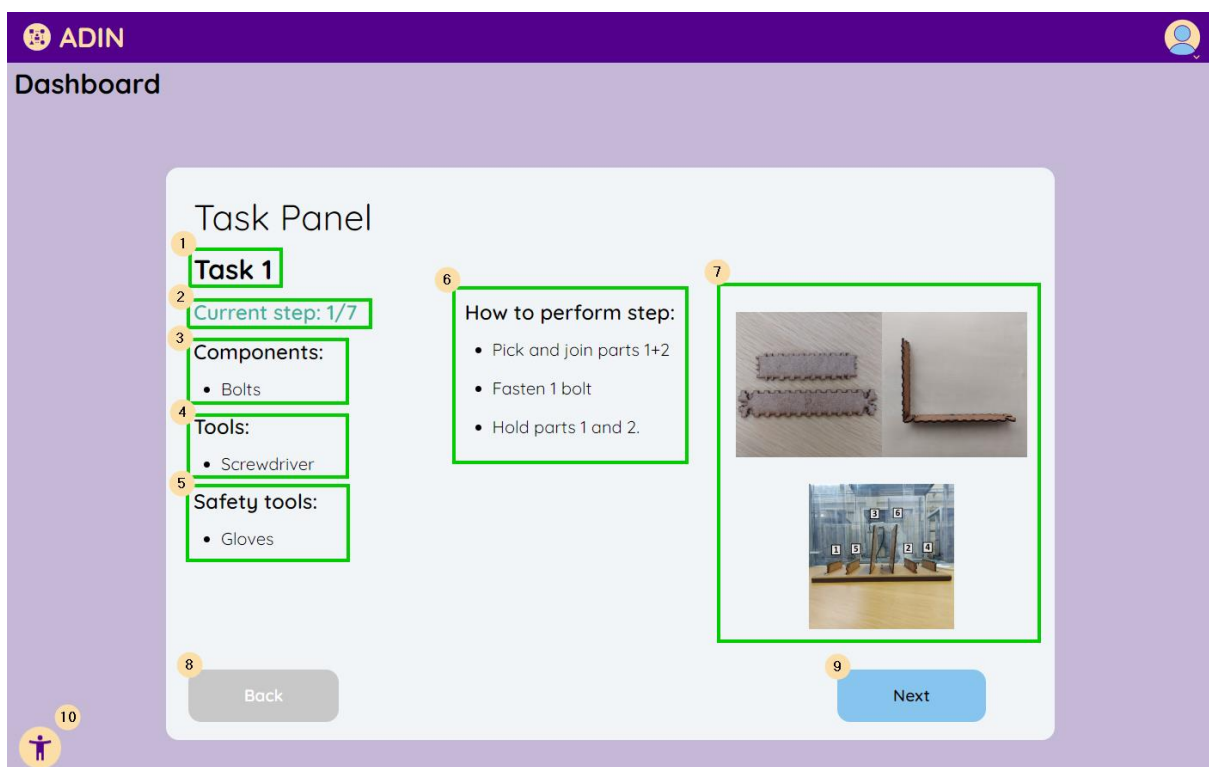
Visual Information:

Back

Aitor Toichoa Eyam
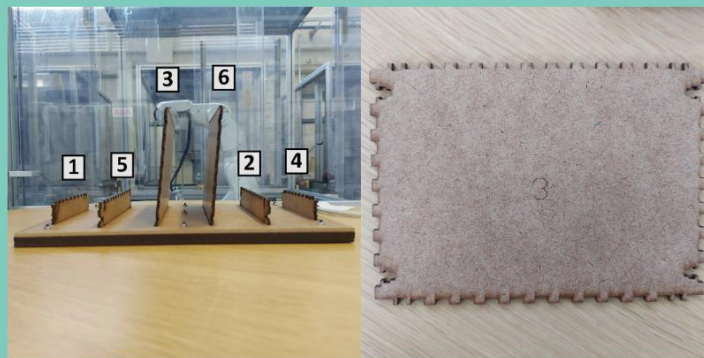
In the following picture, it can be seen all the elements that form the Task Panel.

1. Name of the selected task.
2. Shows the current step that the user is visualizing and how many steps are.

Aitor Toichoa Eyam

3. Necessary components to perform the step.
4. Needed tools to complete the step.
5. Required safety tools to perform the step.
6. Instructions on how to perform the step.
7. Space reserved for pictures related to the step if needed.
8. Button to go see the previous step.
9. Button to proceed to the next step.
10. Accessibility Menu

Once the user is in the last step of the task, by pressing the next button the user is redirected to the first window to select a new task to display.

After logging in, the navigation bar situated on top of the screen shows some different items than before.
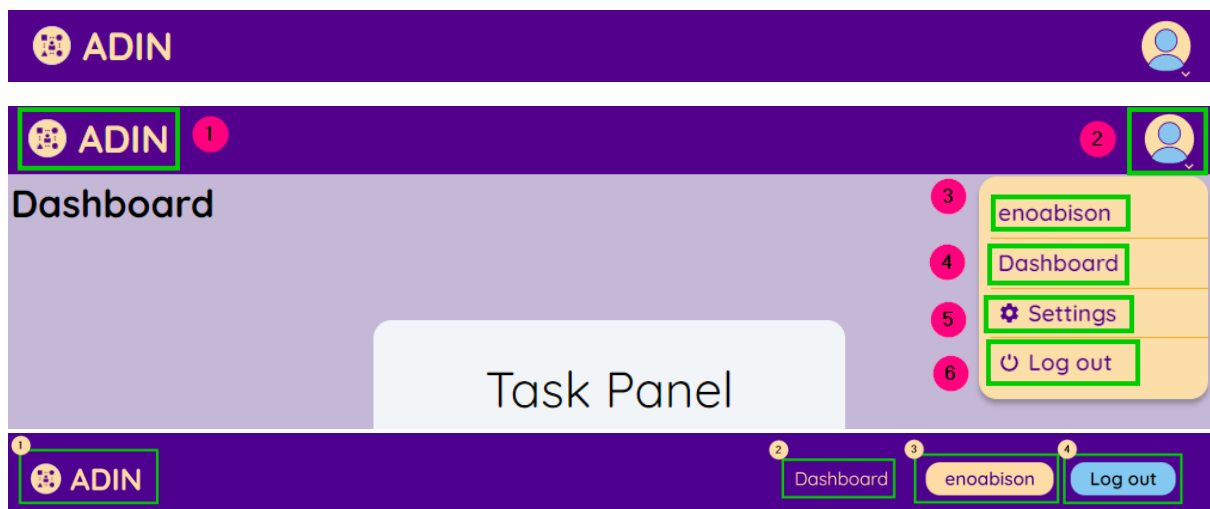
1. ADIN Logo: By pressing the logo the user will be redirected to the home page.
2. User Logged Menu: By pressing the icon, it opens a menu with several options.

Accessibility Menu

The *Accessibility Menu* is opened by pressing the accessibility icon placed in the bottom left corner.
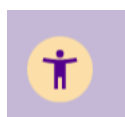
When the users clicks the button, it can be seen several options that the user can modify in order to adapt the application to their personal preference, making it more comfortable to use it whenever they want.
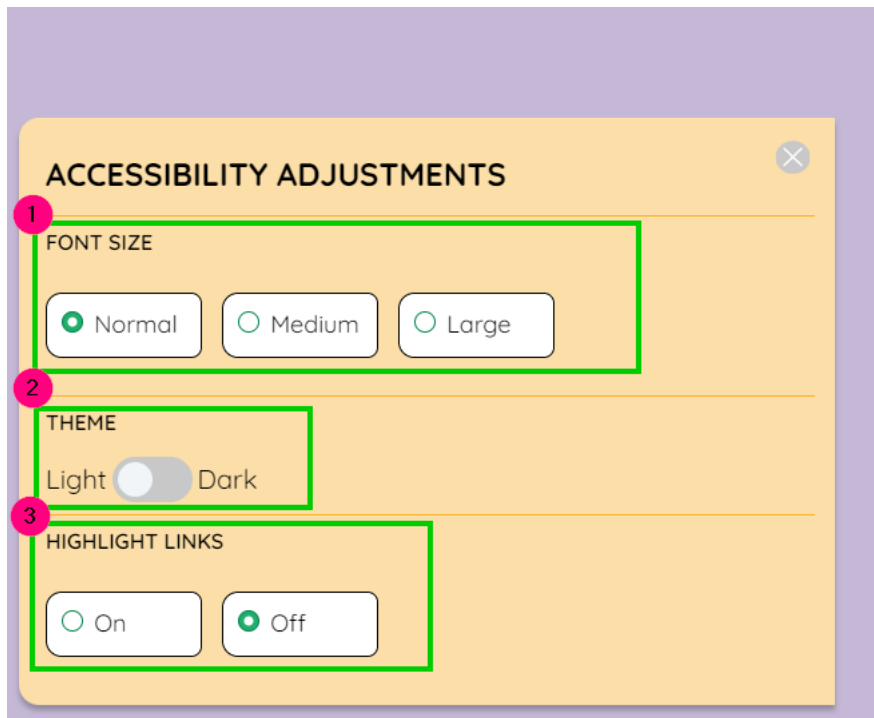
Aitor Toichoa Eyam

1. Font size: The user can increase or decrease the font size of the application.
2. Theme: It changes the theme of the application. There are two options, light or dark theme.
3. Highlight links: By activating this options, all the components of the application that are clickable will have a square surrounding them in order to facilitate the understanding of the elements.

An example of the previous options selected is shown in the following picture. The selected options are *Large* font size, *Dark* theme and *Highlight links* on.

Aitor Toichoa Eyam
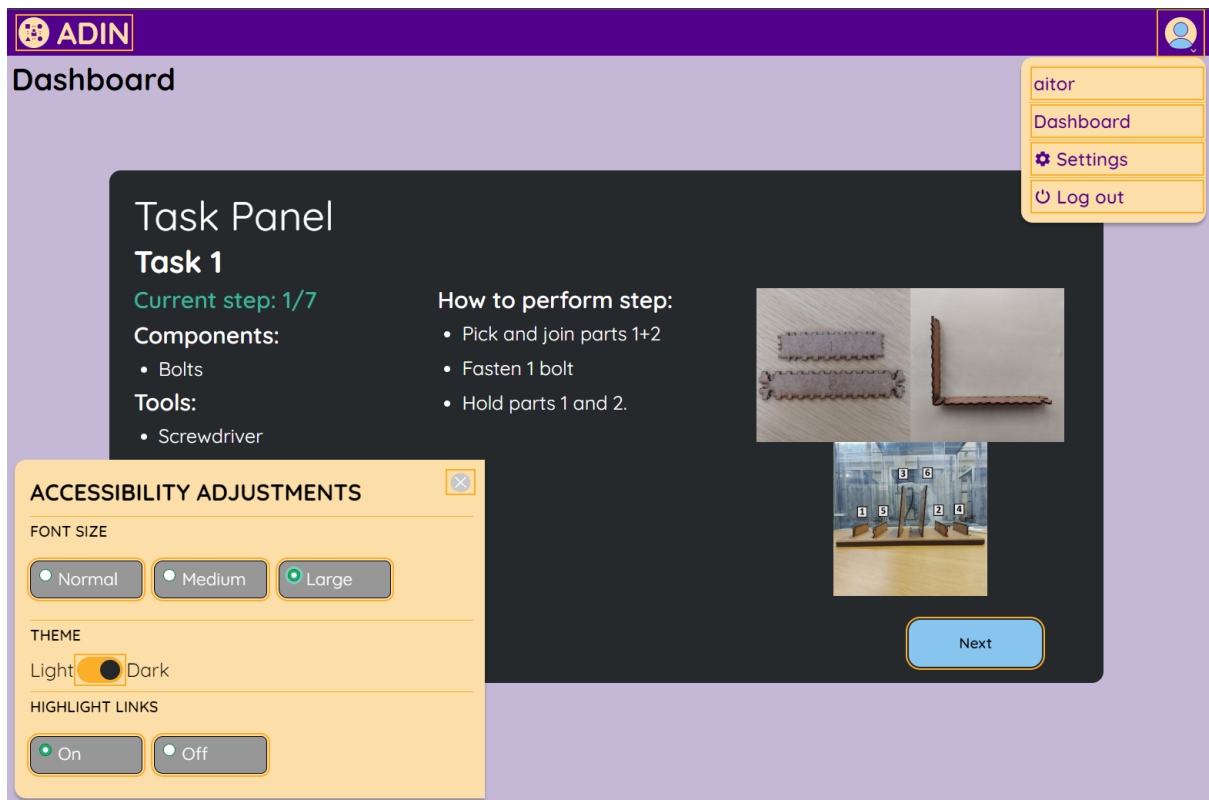
3. Username: This represents the user that has initiated a session in the ADIN application.
4. Dashboard: If the user press this link the application will load the Dashboard page. If the user is already at the Dashboard, it doesn't have any action.
5. Settings: This button opens the Settings page were the use can update the configuration of the application.
6. Log out: Pressing this button will close the session of the user and be automatically redirected to the homepage.

Settings

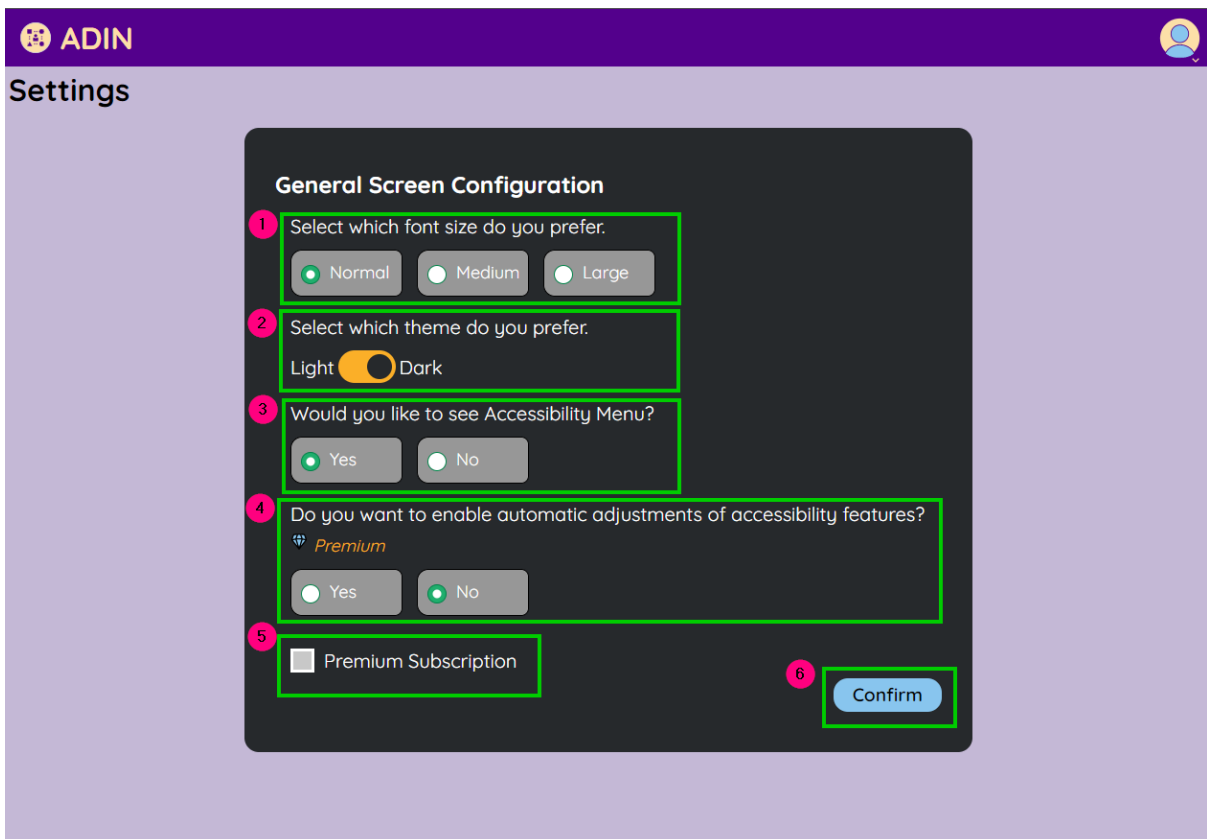In the settings panel, the user can update the configuration of the application and save it in order to have it ready for the next time that logs in.

1. Font size: The user can increase or decrease the font size of the application.
2. Theme: It changes the theme of the application. There are two options, light or dark theme.
3. Accessibility Menu Visibility: It enables or disables the visibility of the Accesibility Menu.
4. Auto adjustments of Accessibility Features: It enables the automatic adjustment of the accessibility features. This feature can only be activated if the user is premium.
5. Premium Subscription: It subscribes the user as premium or not. Allowing or denying the possibility of having premium features.
6. Confirm: Button to save the selected configuration.

Aitor Toichoa Eyam

Small screen

If the screen is reduced, the *Task Panel* will be displayed in another way to fit better the size of the window where the application is being rendered. In the next picture is possible to see the reduced version of the *Task Panel*.

The information shown in the panel is the same but without the additional images. In order to see the information about how to perform the steps, the user should click the info button and a new window will pop up with the information.

Aitor Toichoa Eyam
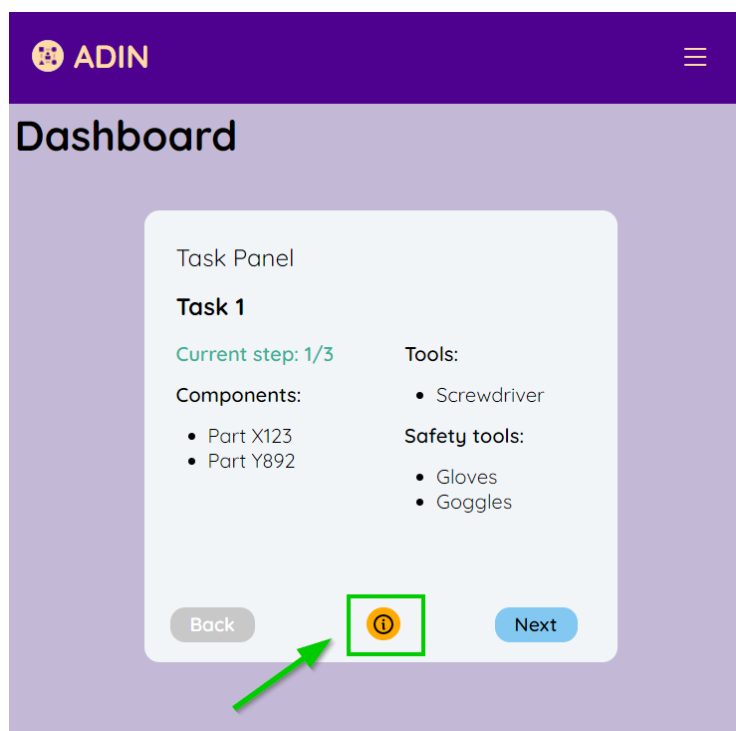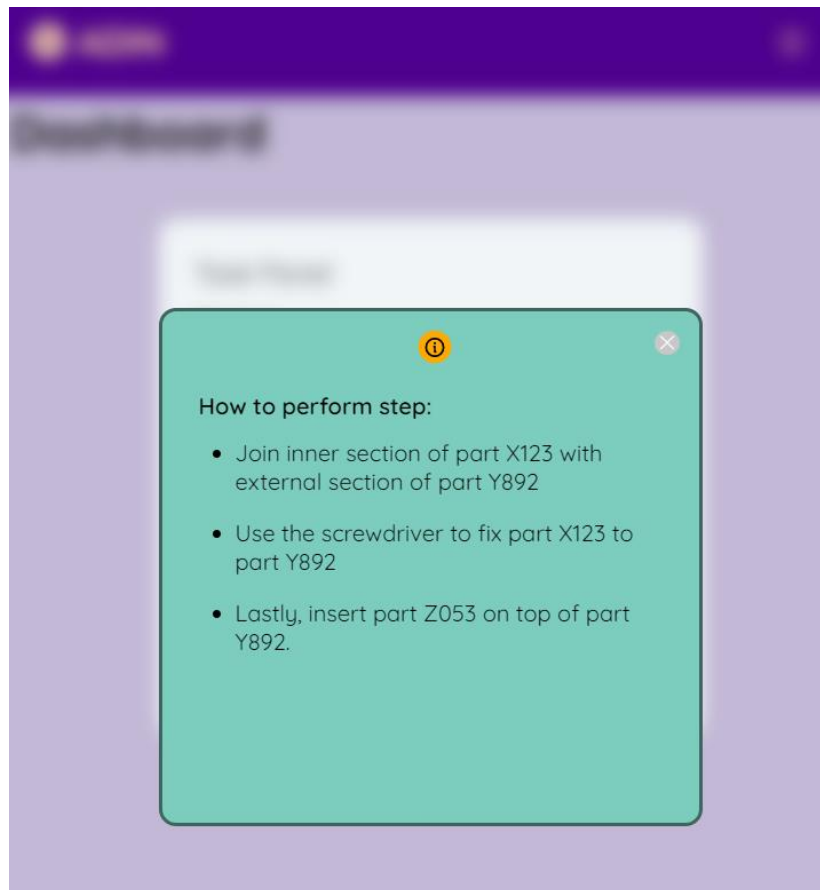
Also, by clicking the hamburger button on the top right corner of the screen, the user would be able to see the items that were displayed before on the navigation bar. The buttons have the same actions that were described before.

Aitor Toichoa Eyam
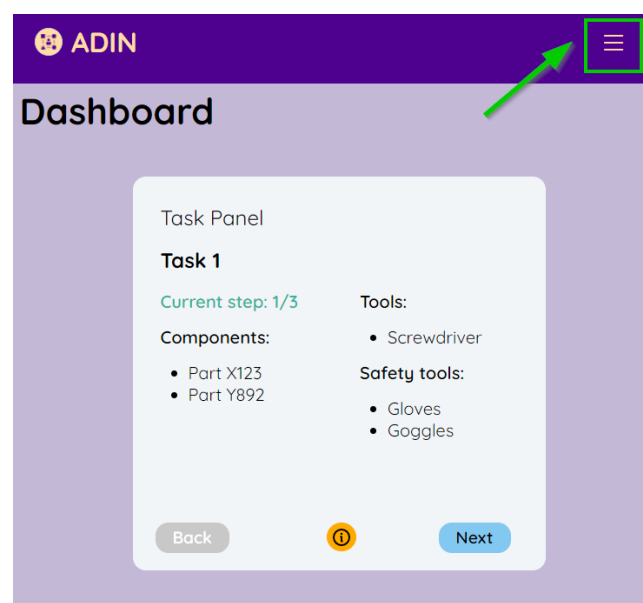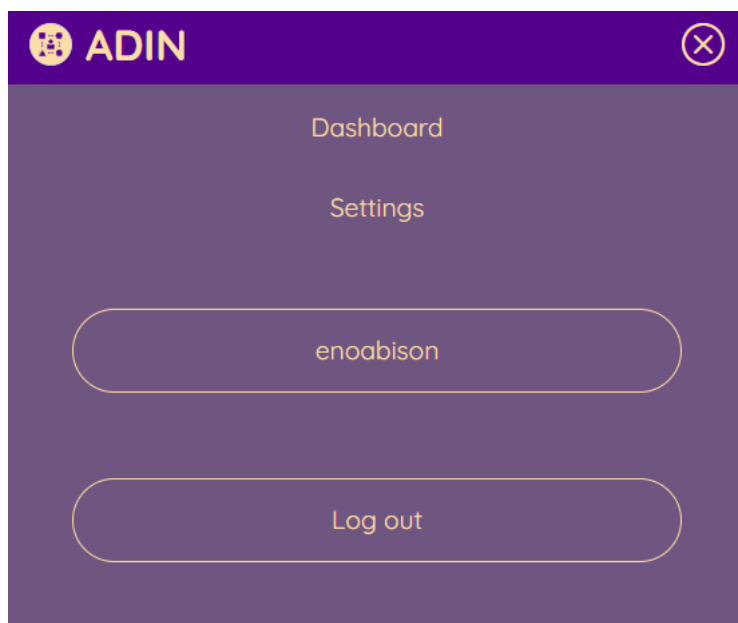
How to structure the database

ADIN shows the information on several tasks in order to assist a user to fulfil them properly. That data is stored in a structured database. This database contains information related to the users, tasks, steps. components, tools and safety tools.

Regarding the data of the user, it is automatically saved after registration. On the other hand, data of the tasks should be introduced manually in the database. It is very important to follow the proper structure of the data so it can be displayed in the application as is meant to be.

In order to have tha database structure configured properly, download the *"adin_db_structure"* file in ADIN's GitHub repository. Once downloaded, use this file as foundation to generate your own PostgreSQL database where you will store all the data that will be used in ADIN. After creating the database, create a folder in the same directory where the docker-compose file is saved and store your database backup there:

```
adin-db:
    image: postgres:14
    environment:
        - POSTGRES_USER=YOUR_USER
        - POSTGRES_PASSWORD=YOUR_PASSWORD
        - POSTGRES_DB=local_adin_db_2
    volumes:
        - pgdata:/var/lib/postgresql/data
        - ./YOUR_DIRECTORY_TO_DATABASE_BACKUP:/docker-entrypoint-initdb.d/init.sql:ro
    ports:
        - "5432:5432"
```

Aitor Toichoa Eyam

Focusing on the creation of tasks is necessary to explain the models, relations among models and tables.

Models

Task:
- task_number: Number of the task. Should be the same as its id.
- description: Task description.

Step:
- task: ForeignKey related to the correspondent task of the step.
- step_number: Number of the step
- instructions: Instructions on how to perform the step.
- components: List of components to use in the step of the assembly.
- tools: List of tools to use in the step of the assembly,
- safety_tools: List of safety tools to use in the step of the assembly.
- images: List of images to show in the step of the assembly.

Component:
- component_name: Name of the component.

Tool:
- tool_name: Name of the tool.

SafetyTool:
- safety_tool_name: Name of the safety tool.

GeneralScreenConfig:
- user: ForeignKey related to the user that has this configuration.
- font_size: Defines the size of the font to be displayed.
- accessibility: Flag marking if user wants to display accessibility features.
- auto_adjust: Flag marking if user wants auto adjustment of accessibility features.
- is_premium: Flag marking if a user is premium or not.

Relations

Task - Step: One to Many
Step - Components: Many to Many
Step - Tools: Many to Many
Step - Safety Tools: Many to Many
Step – Images: Many to Many
User – GeneralScreenConfiguration: One to One

Tables

api_task:
- id: integer (Primary Key(PK))
- task_number: integer
- description: varchar(1000)

api_step:

Aitor Toichoa Eyam

- id: integer
- instructions: varchar(1000)
- step_number: integer
- task_id: bigint (Foreign Key(FK))

api_component:
- id: integer (Primary Key(PK))
- component_name: varchar(50)

api_tool:
- id: integer (Primary Key(PK))
- tool_name: varchar(50)

api_safety_tool:
- id: integer (Primary Key(PK))
- safety_tool_name: varchar(50)

api_step_components:
- id: integer (PK)
- step_id: bigint (FK)
- component_id (FK)

api_step_tools:
- id: integer (PK)
- step_id: bigint (FK)
- safetytool_id (FK)

api_step_safety_tools:
- id: integer (PK)
- step_id: bigint (FK)
- tool_id (FK)

api_step_images:

- id: interger (PK)
- step_id: bigint (FK)
- image_id (FK)

Creation of steps

The tasks and steps information should be as brief and direct as possible. A great number of instructions, components, tools or steps within a task could lead to confusion of the users. As consequence, a reduction of the difficulty of the steps is advisable to have a clear view of the information.

Additionally, for a good representation of the instructions on how to perform the task, these should be separated by points. Example:

Aitor Toichoa Eyam

Join inner section of part X123 with external section of part Y892. Use the screwdriver to fix part X123 to part Y892. Lastly, insert part Z053 on top of part Y892.

id	instructions	step_number	task_id
1	1 Join inner section of part X123 with external section of part Y892. Use the screwdriver to fix part X123 to part ...	1	1

The representation will be as follows:

How to perform step:

- Join inner section of part X123 with external section of part Y892

- Use the screwdriver to fix part X123 to part Y892

- Lastly, insert part Z053 on top of part Y892.

Aitor Toichoa Eyam