

ADIN

(ADAPTIVE INTERFACES)

Make adaptability your main power

User Manual



Contents

Introduction	2
Component Access.....	3
Software requirements.....	3
Set up	3
Application Sections.....	4
How to structure the database	14

Introduction

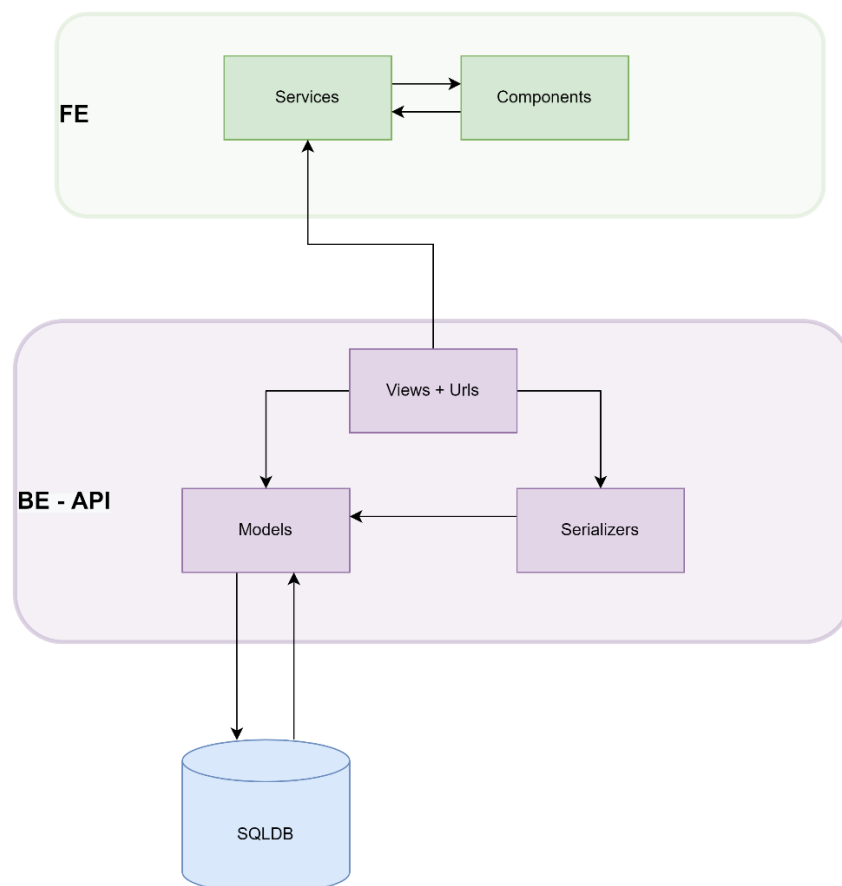
The purpose of this document is explain the user what is tof the application, how to set it up and show how to use its main features.

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 three 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
- Database (SQLDB): It is where is stored all the necessary data that the application needs for its correct operation.



Component Access

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

The component is divided into two artifacts. In order to use ADIN, it is necessary to download both artifacts:

- *adin-api*
- *adin-fe*

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 advisable 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 both artifacts, *adin-api* and *adin-fe* from RAMP. In the next section is explained how to run the application using docker compose.

Docker Compose approach

Docker-compose requires having both artifacts in a common folder with full rights. Inside the folder containing both artifacts create a YAML file named *docker-compose*. This file can be generated in a text editor such as *Notepad++* or similar. Create the file, write in the document the following commands and and save it with the extension *.yaml*:

```
version: "3"
services:
  adin-api:
    image: bnaito/adin-api:test
    container_name: adin-api
    command: bash -c "python manage.py runserver 0.0.0.0:8000"
    environment:
      - discovery.type=single-node
    build:
      context: ./adin-be/
    ports:
      - 8000:8000
  adin-fe:
    image: bnaito/adin-fe:test
    container_name: adin-fe
    command: npm start
    environment:
      - discovery.type=single-node
    build:
      context: ./adin-fe/
    depends_on:
      - adin-api
    ports:
      - 3000:3000
```

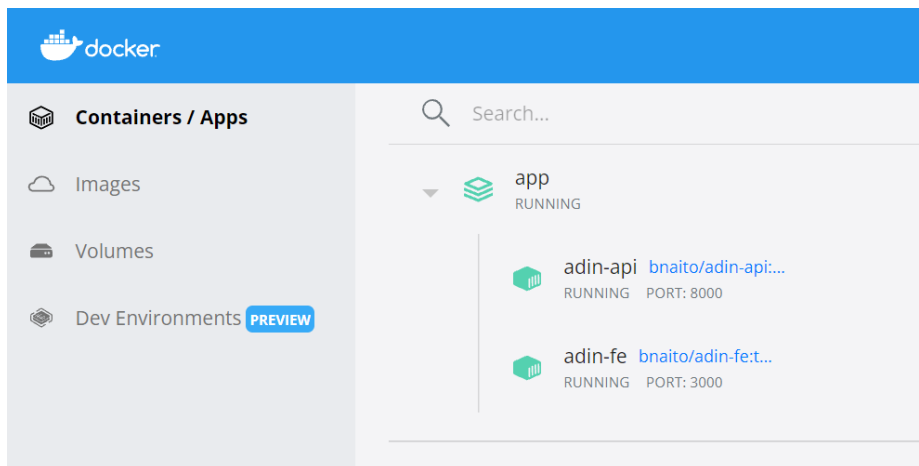
The folder should have the following elements:

-  `adin-fe`
-  `adin-be`
-  `docker-compose`

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 `adin-fe`, `adin-be` and `docker-compose.yml` are and execute the command: **`docker-compose up`**.

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

To check if the application has been deployed correctly, check at Docker Desktop the *Containers/App* section:



To stop the application, use the command: **`docker-compose down -v`**, it will destroy the cluster and the data volumes.

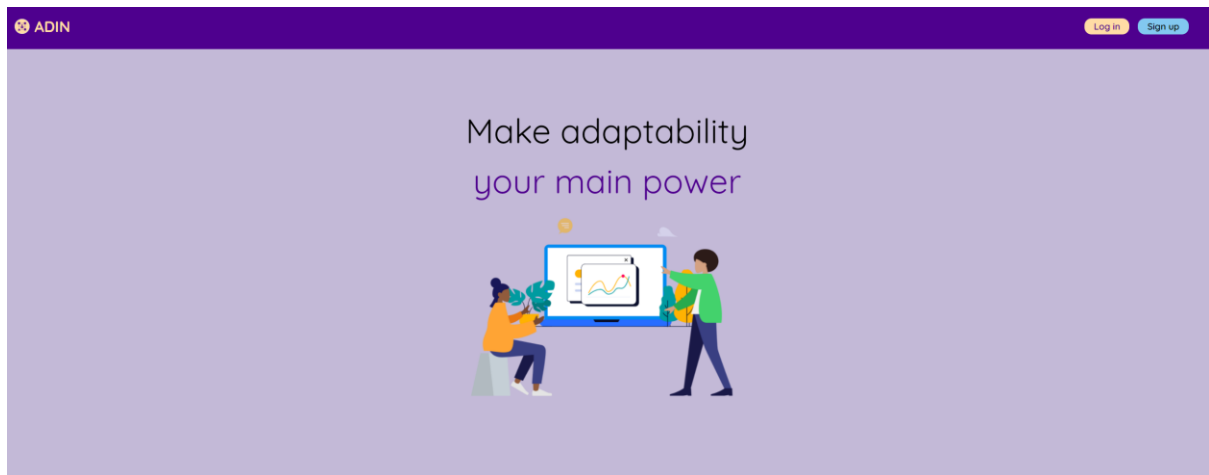
Application Sections

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

Home Page

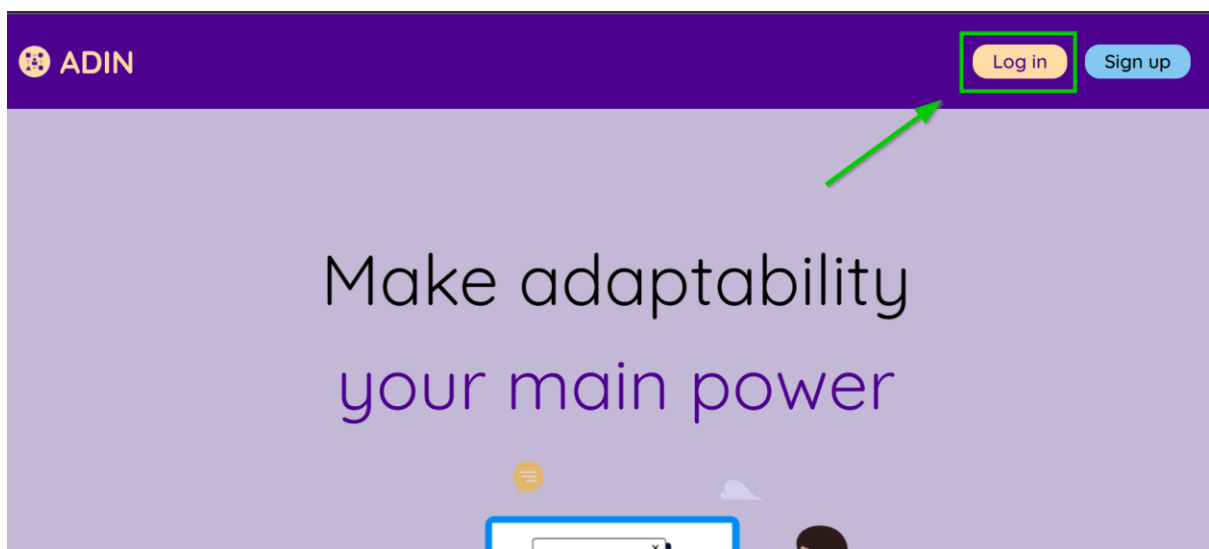
This is the first page of the application where a user can perform to main actions:

- 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.

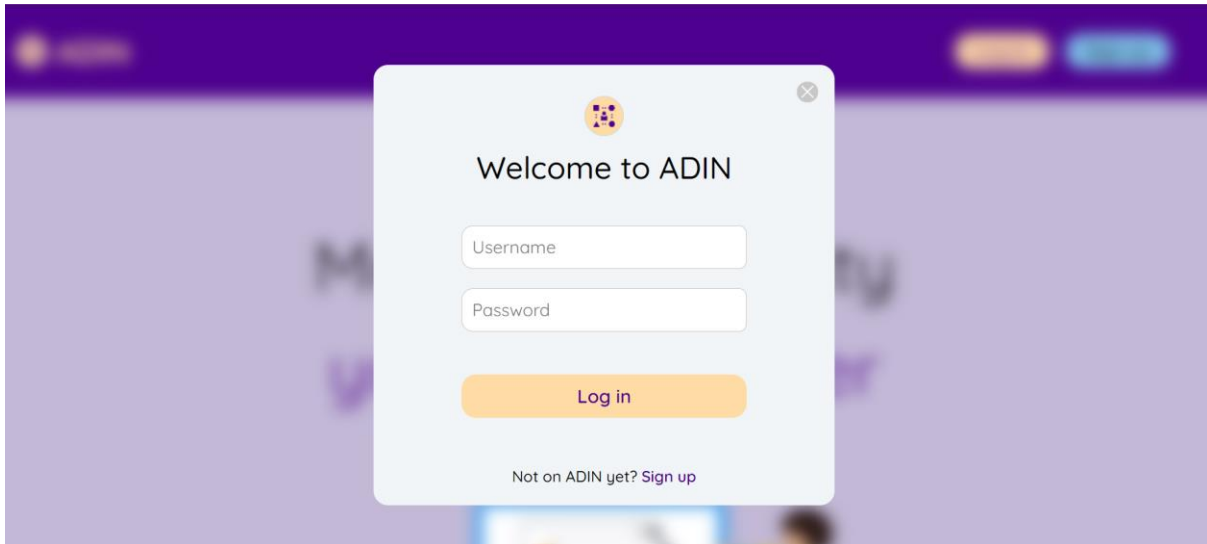


Log in

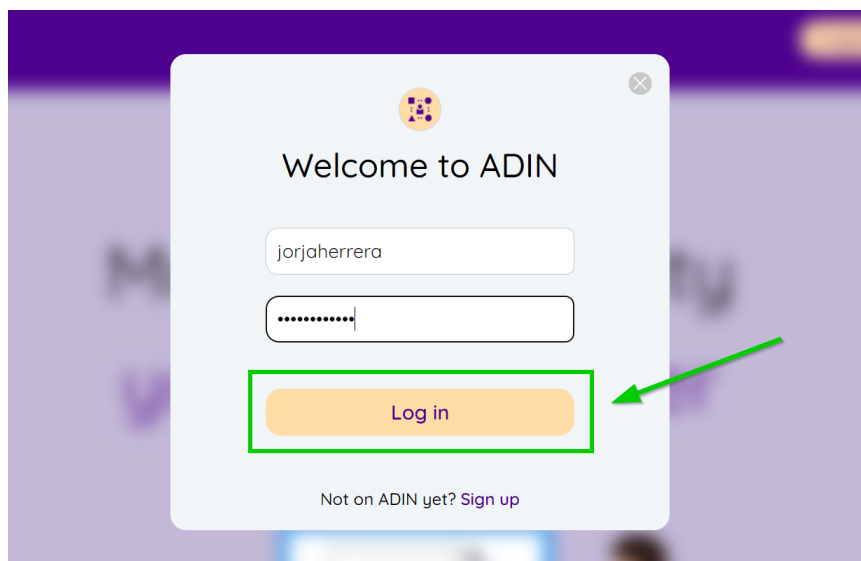
To log in to the application the user should press the *log in* button which is located in the upper right section of the window.



After pressing the button, it will appear a popup window in which the user should fill two fields, username and password.

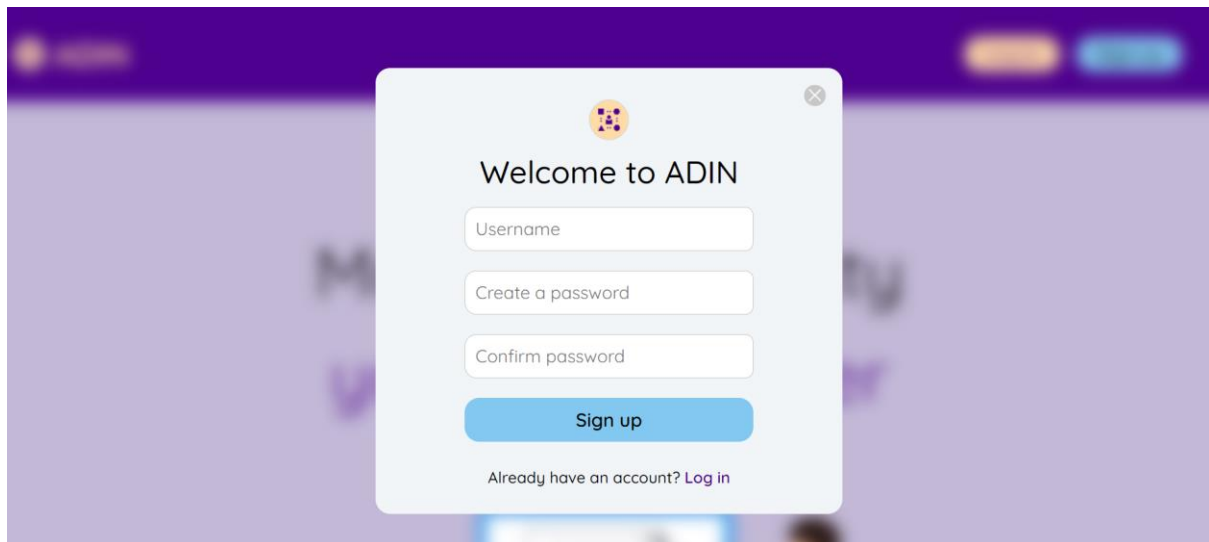
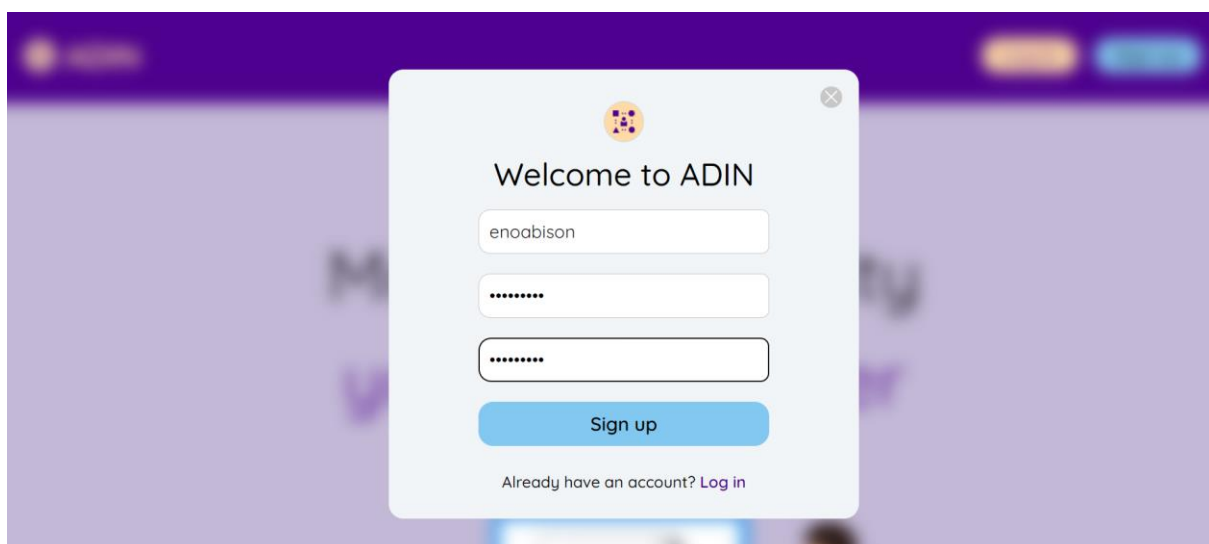


Once the username and password have been introduced, by pressing the *Log in* button of the popup window, the user will be redirected to the Dashboard.

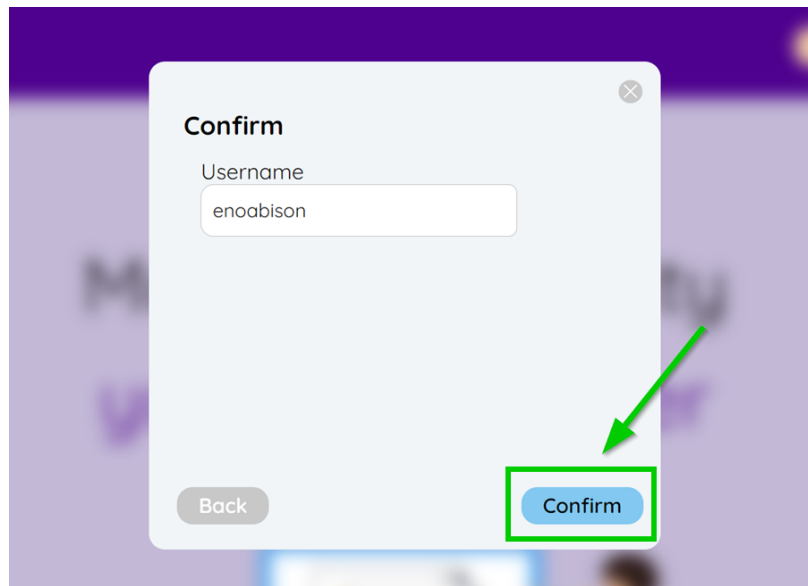


Sign up

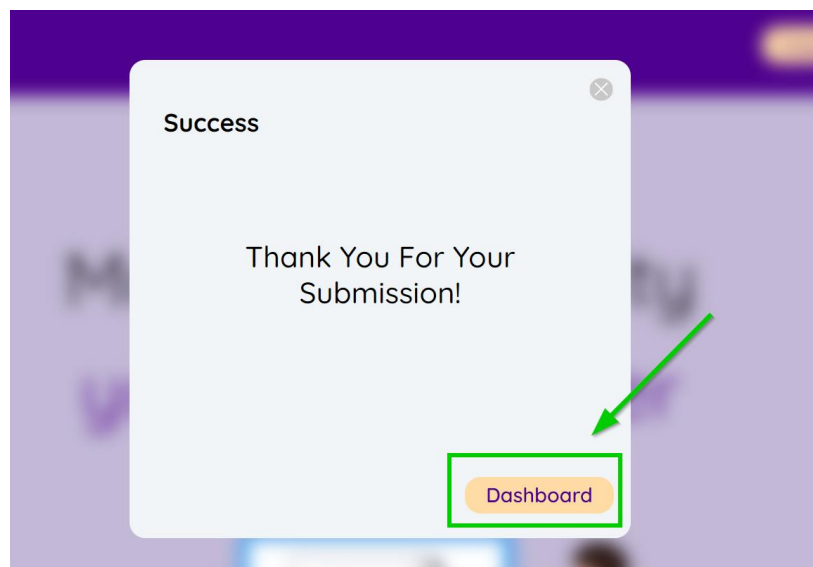
Similar to the login process, in order to sign up in the application, the user should press the *sign up* button located in the top right corner of the screen. A popup window will emerge asking the user to introduce data to generate a profile.

A screenshot of a web application showing a registration modal titled "Welcome to ADIN". The modal has a light blue background and a close button in the top right corner. It contains four input fields: "Username", "Create a password", and "Confirm password", followed by a blue "Sign up" button. At the bottom, there is a link "Already have an account? Log in". The background of the page is a blurred purple and blue interface.A screenshot of the same registration modal, now with data entered. The "Username" field contains "enoabison". The "Create a password" and "Confirm password" fields are filled with masked characters (dots). The "Sign up" button remains blue. The "Log in" link is still present at the bottom. The background is the same blurred interface.

Once the fields are filled, the user 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.

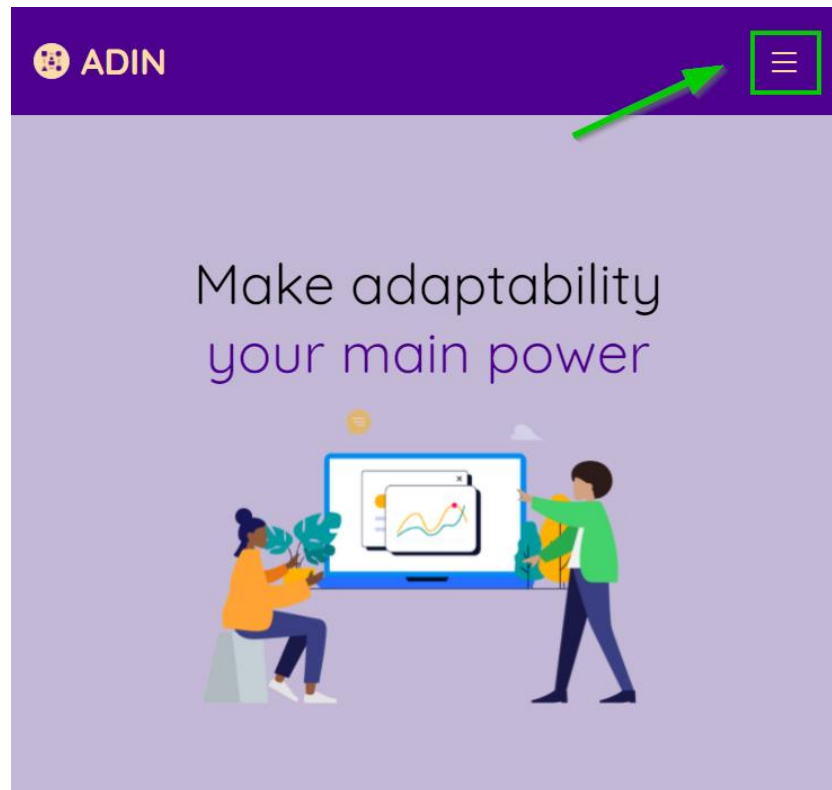


Pressing the *Dashboard* button will lead the user to the Dashboard.

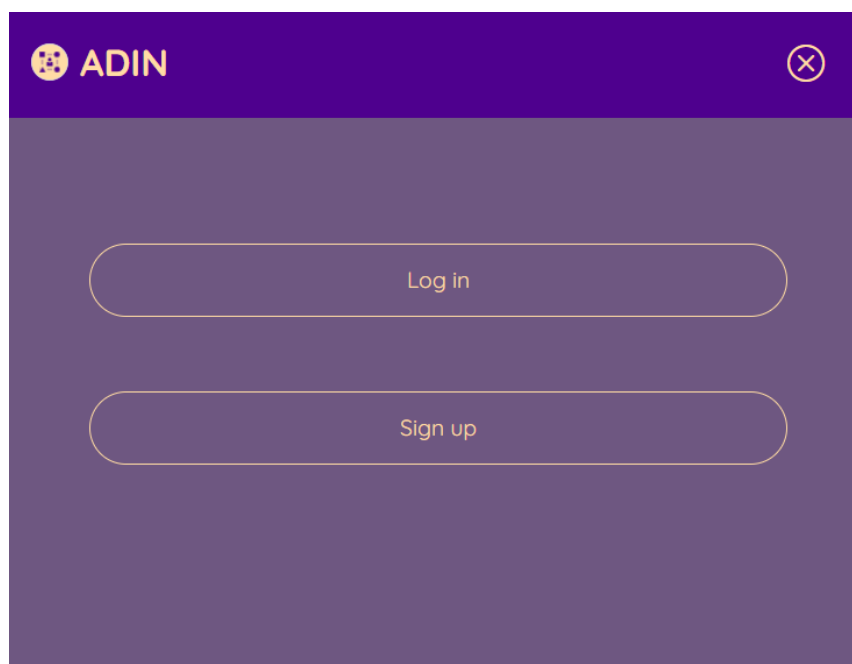


Small screen

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.



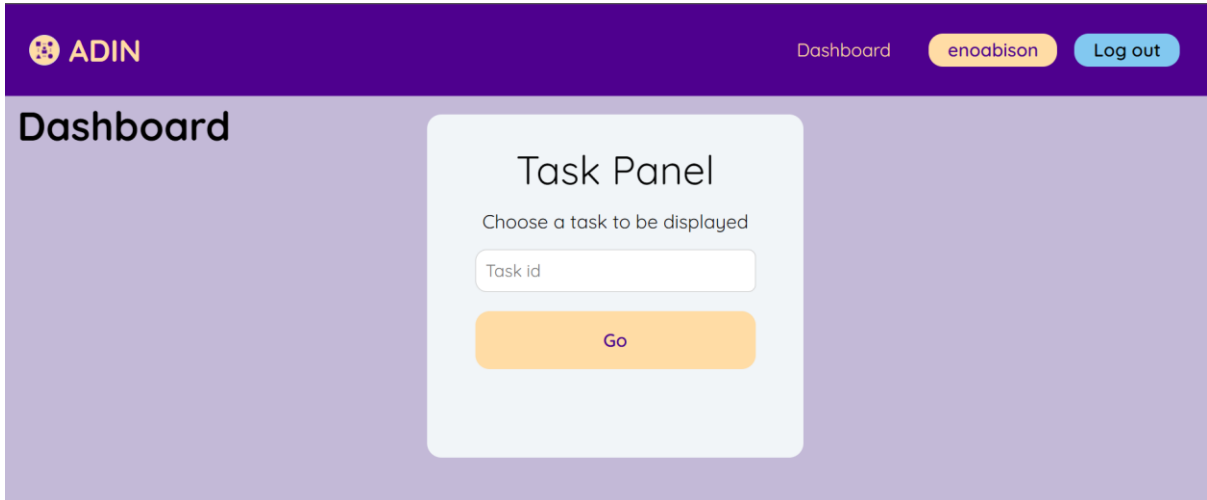
By clicking the hamburger icon, a menu will be displayed on the screen with the same buttons that were contained in the navigation bar. The user can make the same procedures of login and sign up described before.



Dashboard

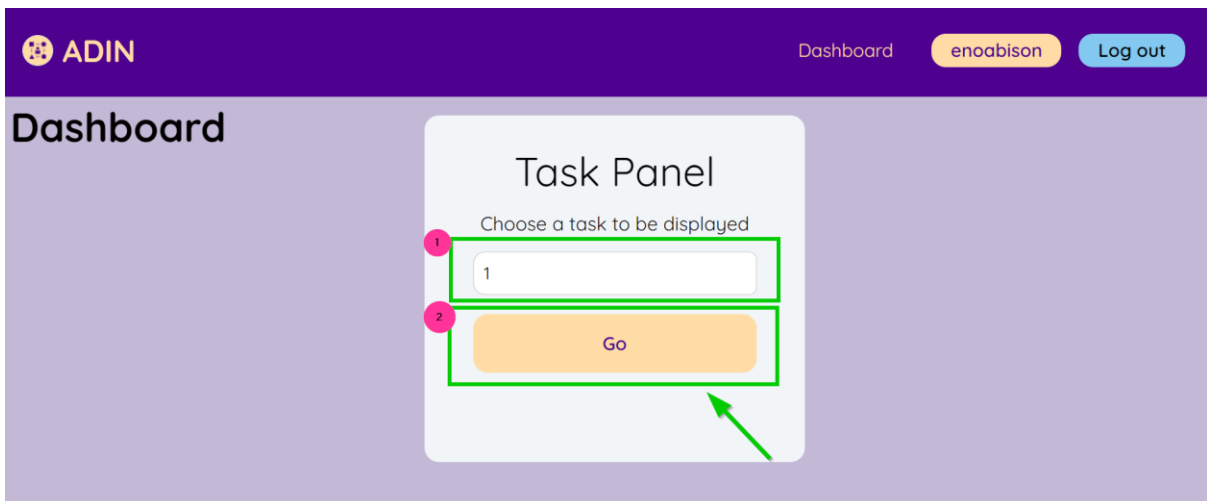
On this page, it is represented the *Task Panel*. The *Task Panel* is meant to assist the user through the development of a task by showing the required steps and information to fulfil without any risk the selected task.

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

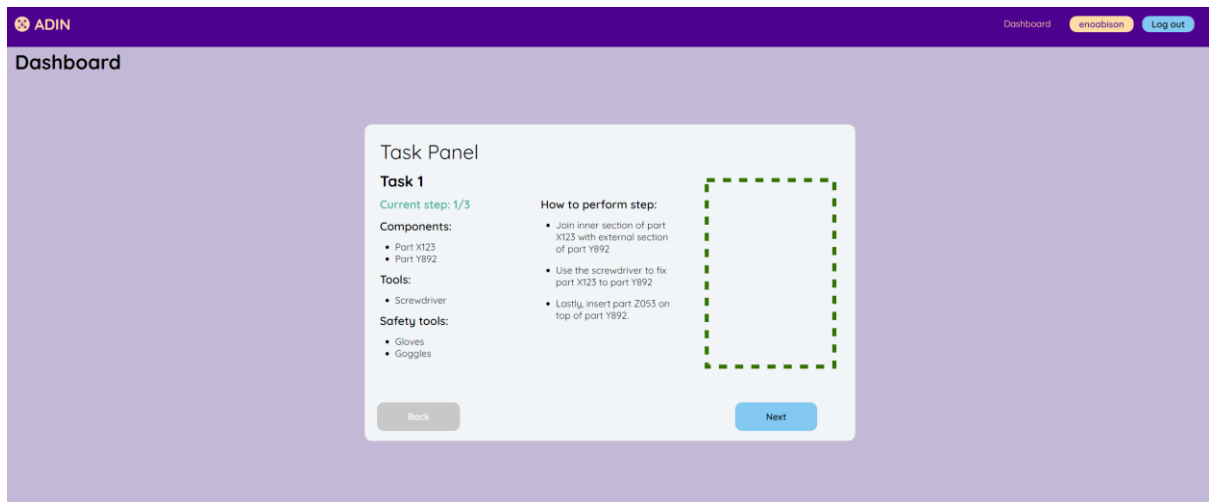


The screenshot shows the ADIN Dashboard interface. At the top, there is a purple header bar with the ADIN logo on the left, and the text "Dashboard", "enoabison", and "Log out" on the right. Below the header, the word "Dashboard" is displayed in large black text. In the center, there is a light blue rounded rectangle titled "Task Panel". Inside this panel, the text "Choose a task to be displayed" is shown above a white input field labeled "Task id". Below the input field is an orange button with the text "Go".

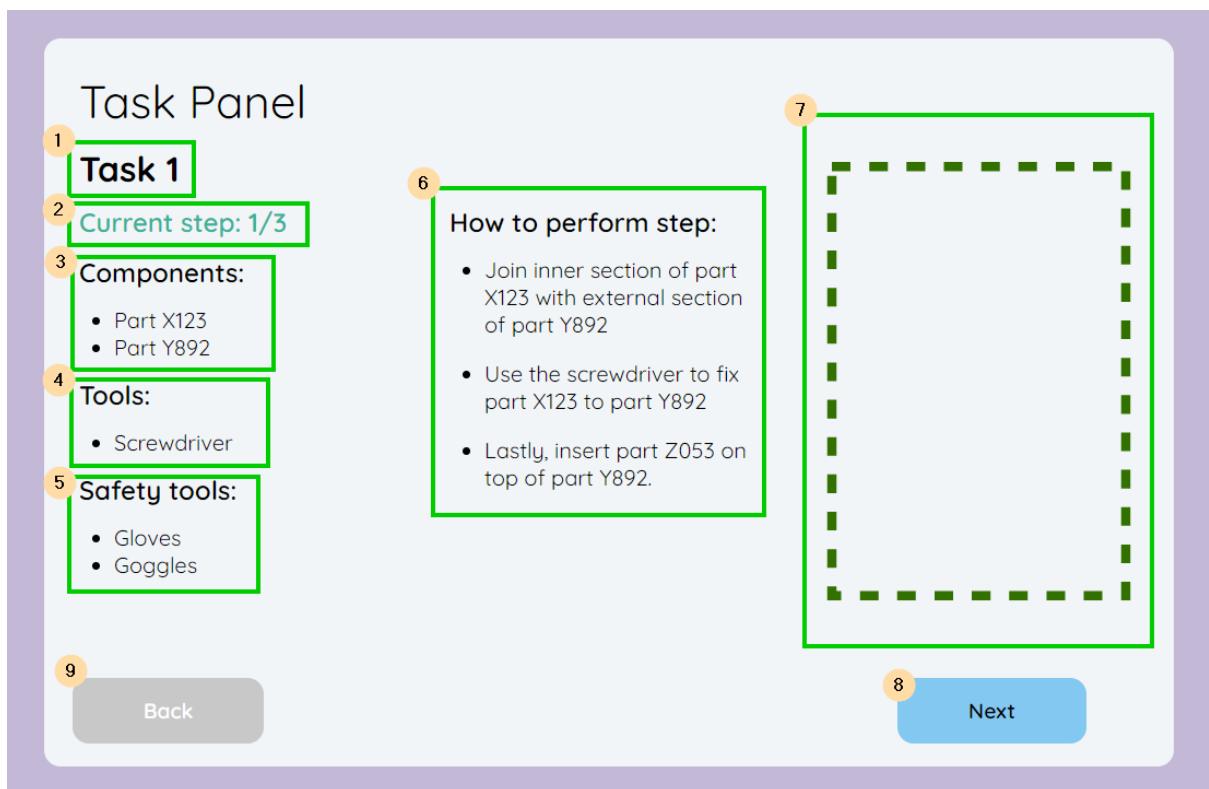
By introducing the id number of the task and pressing the *Go* button the application shows the *Task Panel* with the information of the selected task.



This screenshot is identical to the previous one, but with green annotations. A green rectangle highlights the "Task id" input field, which contains the number "1". A pink circle with the number "1" is placed to the left of this field. Another green rectangle highlights the "Go" button. A pink circle with the number "2" is placed to the left of this button. A green arrow points from the bottom right towards the "Go" button.



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.
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 proceed to the next step.
9. Button to go see the previous step.

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.

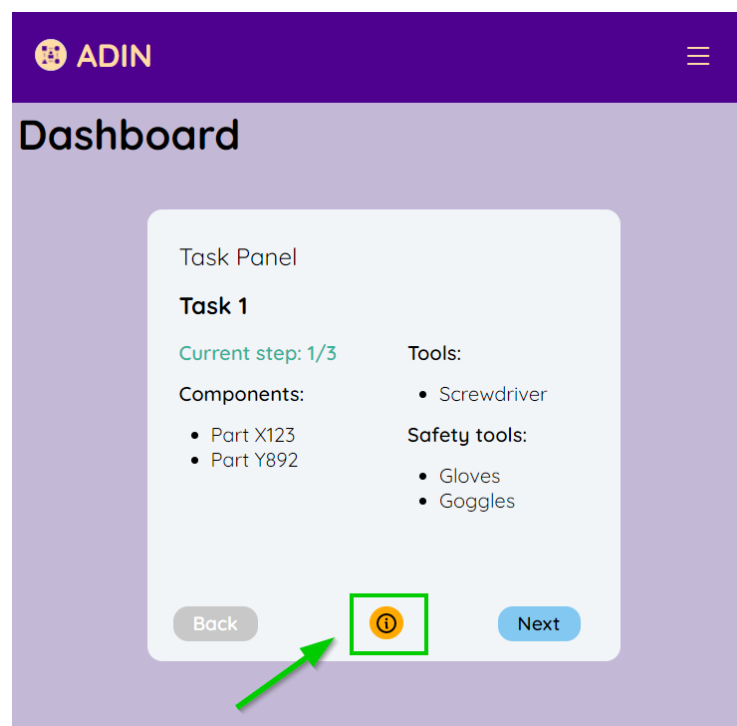


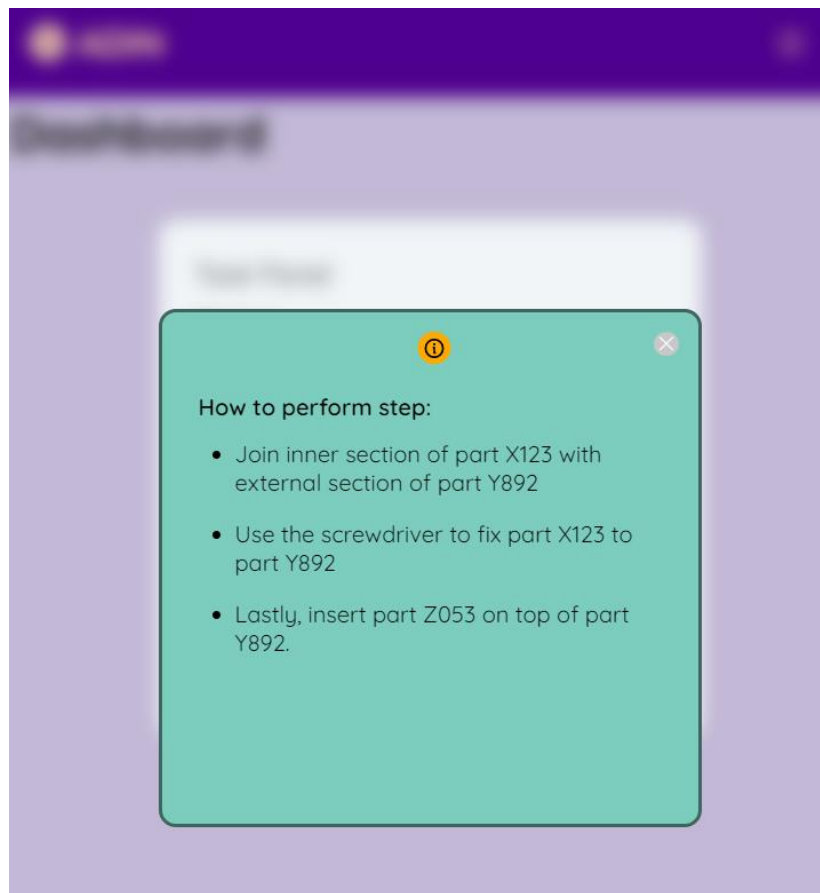
1. ADIN Logo: By pressing the logo the user will be redirected to the home page.
2. Dashboard: If the user press this link the application will load the Dashboard page. If the user is already at the Dashboard, it doesn't have any action.
3. Username: This represents the user that has initiated a session in the ADIN application.
4. Log out: Pressing this button will close the session of the user and be automatically redirected to the homepage.

Small screen

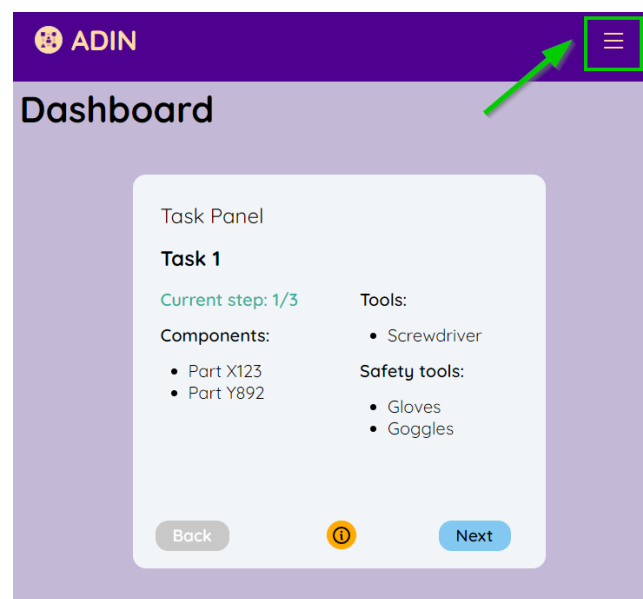
If the screen is reduced, the *Task Panel* will be displayed in another way to fit better the size of the window where the application is being rendered. In the next picture is possible to see the reduced version of the *Task Panel*.

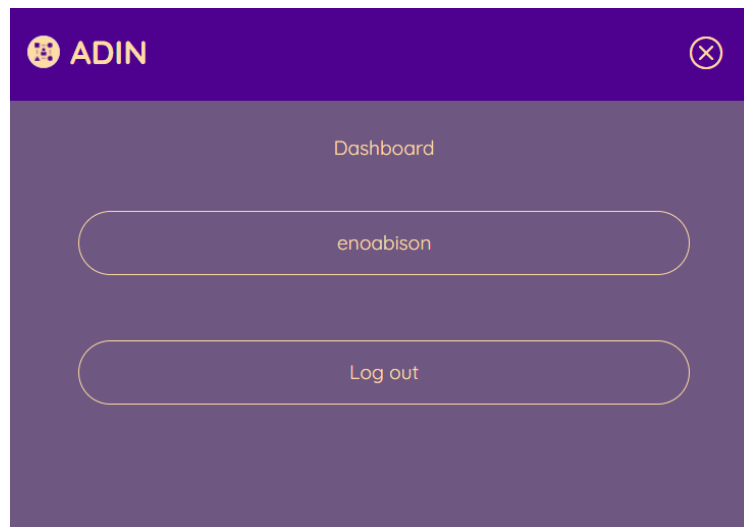
The information shown in the panel is the same but without the additional images. In order to see the information about how to perform the steps, the user should click the info button and a new window will pop up with the information.





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.





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.

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.

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.

Component:

- component_name: Name of the component.

Tool:

- tool_name: Name of the tool.

SafetyTool:

- safety_tool_name: Name of the safety tool.

Relations

Task - Step: One to Many

Step - Components: Many to Many

Step - Tools: Many to Many

Step - Safety Tools: Many to Many

Tables

Task:

- id: integer (Primary Key(PK))
- task_number: integer

Step:

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

Component:

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

Tool:

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

Safety_Tool:

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

Step_Components:

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

Step_Tools:

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

Step_Safety_Tools:

- id: integer (PK)
- step_id: bigint (FK)
- tool_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:

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.