INFORMATICS FACULTY

T120B165 Web Application Design

Project report

Student: Rapolas Kutka
Lecturer: Tomas Blažauskas

2023

1.1. Purpose of the system

The aim of the project is to help end users to save time and efficiently use laundromat and to help administrator with equipment maintenance. Using this system registered users will be able to reserve washing machine or any other type of equipment for specified time period. If machine malfunctions users will have opportunity to register problem and administrator will see it. Administrator can see users history, change/see machines status and delete user if necessary.

1.2. Functional requirements

Unregistered system user can:

1. View the platform representative page;
2. Log in to the online application.

A *registered* system user can:

1. Disconnect from the online application;
2. Log in (register) to the platform;
3. Reserve a machine:

3.1. Select machine type;

3.2. See available time for reservation;

3.3. Reserve time slot for certain machine;

4. Report problem with equipment;

5. Report lost and found items for other users;

The *administrator* can:

1. View users machine usage history.
2. Add new machine.
3. Remove users.
4. Remove or disable machine.

1.3. System architecture

System components:

- **Client side** (FrontEnd) – will use Vue.js;
- **Server side** (Back-End) - will use PHP Symfony. Database - MySQL.

In the Figure 1. the deployment diagram of the system under development is shown. System will be hosted on linux server. Every part of the system will be installed on the same server. The web application is accessed via the HTTP protocol. The operation of this system (eg. data manipulation with the database) requires the laundromat API, which is available through the application programming interface. The laundromat API itself exchanges data with the database - the ORM is used for this purpose.

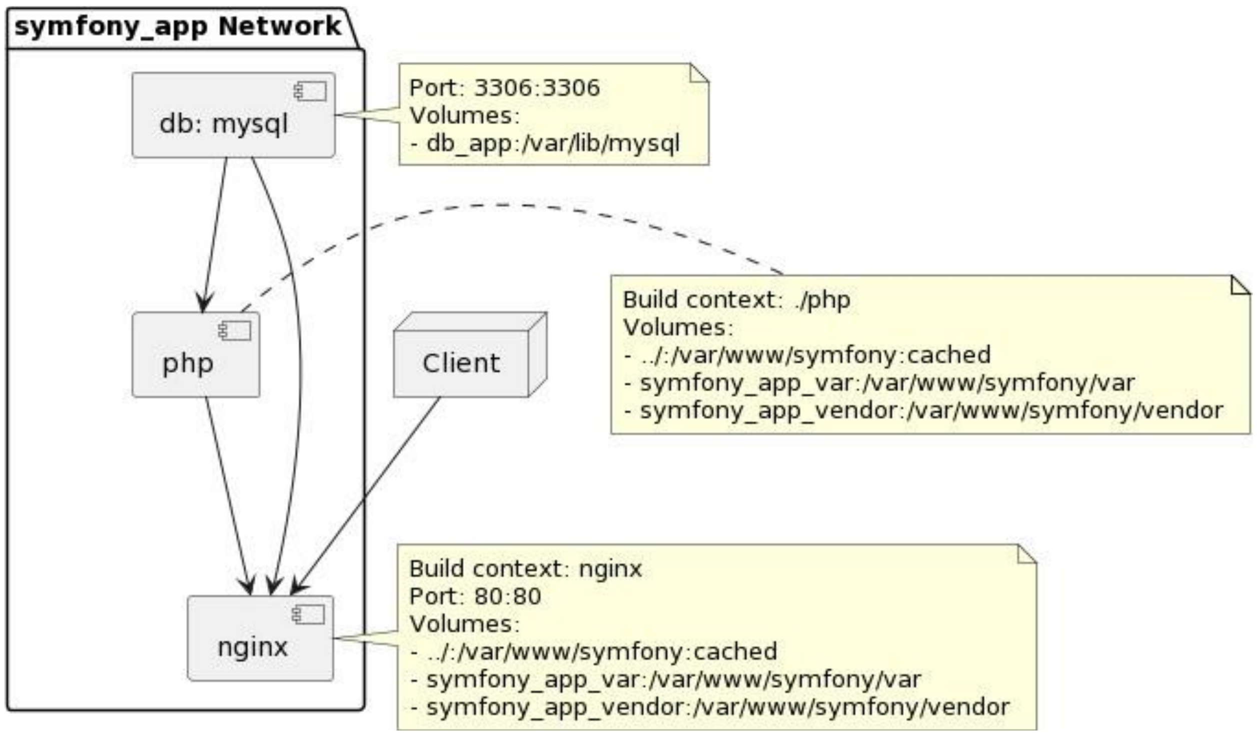

symfony_app Network
db: mysql
php
nginx
Client
Port: 3306:3306
Volumes:
- db_app:/var/lib/mysql
Build context: ./php
Volumes:
- ../:/var/www/symfony:cached
- symfony_app_var:/var/www/symfony/var
- symfony_app_vendor:/var/www/symfony/vendor
Build context: nginx
Port: 80:80
Volumes:
- ../:/var/www/symfony:cached
- symfony_app_var:/var/www/symfony/var
- symfony_app_vendor:/var/www/symfony/vendor

Figure 1. Laundromat system deployment diagram

API specification

API methods for users:

API method	**User index GET**
Purpose	**Get user list**
Route	**/api/users**
Request structure	**-**
Header	**Authorization: Bearer {token}**

Response structure	**[
	  {
	    "id": 0,
	    "email": "string",
	    "roles": [
	      "string"
	    ],
	    "dorm": "../example",
	    "history": [
	      "../example"**

	] }]
Response code	200 (OK)

API method	Create user POST
Purpose	Create user
Route	/api/users

Request structure	{
	"email": "string",
	"roles": [
	"string"
	],
	"password": "string",
	"dorm": "../example",
	"history": [
	"../example"
	]
	}

Header	Authorization: Bearer {token}

Response structure	{
	"id": 0,
	"email": "string",
	"roles": [
	"string"
	],
	"dorm": "../example",
	"history": [
	"../example"
	]
	}

Response code	201 (OK)

API method	Show user GET
Purpose	Show user
Route	/api/users/{id}
Request structure	-
Header	Authorization: Bearer {token}
Response structure	{ "id": 0,

	"email": "string", "roles": ["string"], "dorm": "../example", "history": ["../example"] }
Response code	200 (OK)

API method	Edit user PATCH
Purpose	To change users information
Route	/api/users/{id}
Request structure	{ "email": "string", "roles": ["string"], "password": "string", "dorm": "../example", "history": ["../example"] }
Header	Authorization: Bearer {token}
Response structure	{ "id": 0, "email": "string", "roles": ["string"], "dorm": "../example", "history": ["../example"] }
Response code	200 (OK)

API method	Remove user DELETE

Purpose	**Delete user**
Route	**/api/users/{id}**
Request structure	**-**
Header	**Authorization: Bearer {token}**
Response structure	**-**
Response code	**204 (OK)**

API methods for dorms:

API method	**Dorm index GET**
Purpose	**Get dorm list**
Route	**/api/dorms**
Request structure	**-**
Header	**Authorization: Bearer {token}**
Response structure	**[{ "id": 0, "name": "string", "machines": ["../example"], "administrator": "../example", "residents": ["../example"] }]**
Response code	**200 (OK)**

API method	**Create dorm POST**
Purpose	**Create dorm**
Route	**/api/dorms**
Request structure	**{ "name": "string", "machines": ["../example"], "administrator": "../example", "residents": ["../example"**

	] }
Header	Authorization: Bearer {token}

Response structure	{
	  "id": 0,
	  "name": "string",
	  "machines": [
	    "../example"
	  ],
	  "administrator": "../example",
	  "residents": [
	    "../example"
	  ]
	}

Response code	201 (OK)

API method	Show dorm GET
Purpose	Show dorm
Route	/api/dorms/{id}
Request structure	-
Header	Authorization: Bearer {token}

Response structure	{
	  "id": 0,
	  "name": "string",
	  "machines": [
	    "../example"
	  ],
	  "administrator": "../example",
	  "residents": [
	    "../example"
	  ]
	}

Response code	200 (OK)

API method	Edit dorm PATCH
Purpose	To change dorm information
Route	/api/dorms/{id}

Request structure	{
	  "name": "string",
	  "machines": [
	    "../example"

	], "administrator": "../example", "residents": ["../example"] }
Header	**Authorization: Bearer {token}**
Response structure	**{ "id": 0, "name": "string", "machines": ["../example"], "administrator": "../example", "residents": ["../example"] }**
Response code	**200 (OK)**

API method	**Remove dorm DELETE**
Purpose	**Delete dorm**
Route	**/api/dorms/{id}**
Request structure	**-**
Header	**Authorization: Bearer {token}**
Response structure	**-**
Response code	**204 (OK)**

API methods for machine:

API method	**Machine index GET**
Purpose	**Get machine list**
Route	**/api/machines**
Request structure	**-**
Header	**Authorization: Bearer {token}**
Response structure	**[{ "id": 0, "type": "string", "isActive": true,**

	"lastMaintenance": "2019-08-24T14:15:22Z", "dorm": "../example", "history": ["../example"] }]
Response code	**200 (OK)**

API method	**Create machine POST**
Purpose	**Create machine**
Route	**/api/machines**

Request structure	{
	"type": "string",
	"isActive": true,
	"lastMaintenance": "2019-08-24T14:15:22Z",
	"dorm": "../example",
	"history": [
	"../example"
	]
	}

Header	**Authorization: Bearer {token}**

Response structure	{
	"id": 0,
	"type": "string",
	"isActive": true,
	"lastMaintenance": "2019-08-24T14:15:22Z",
	"dorm": "../example",
	"history": [
	"../example"
	]
	}

Response code	**201 (OK)**

API method	**Show machine GET**
Purpose	**Show machine**
Route	**/api/machines/{id}**

Request structure	-
Header	Authorization: Bearer {token}

Response structure	{
	  "id": 0,
	  "type": "string",
	  "isActive": true,
	  "lastMaintenance": "2019-08-24T14:15:22Z",
	  "dorm": "../example",
	  "history": [
	    "../example"
	  ]
	}

Response code	200 (OK)

API method	Edit machine PATCH
Purpose	To change machine information
Route	/api/machines/{id}

Request structure	{
	  "type": "string",
	  "isActive": true,
	  "lastMaintenance": "2019-08-24T14:15:22Z",
	  "dorm": "../example",
	  "history": [
	    "../example"
	  ]
	}

Header	Authorization: Bearer {token}

Response structure	{
	  "id": 0,
	  "type": "string",
	  "isActive": true,
	  "lastMaintenance": "2019-08-24T14:15:22Z",
	  "dorm": "../example",
	  "history": [
	    "../example"
	  ]
	}

Response code	200 (OK)

API method	**Remove machine DELETE**
Purpose	**Delete machine**
Route	**/api/machines/{id}**
Request structure	**-**
Header	**Authorization: Bearer {token}**
Response structure	**-**
Response code	**204 (OK)**

API methods for history entries:

API method	**History index GET**
Purpose	**Get history list**
Route	**/api/histories**
Request structure	**-**
Header	**Authorization: Bearer {token}**
Response structure	**[{ "id": 0, "startDate": "2019-08-24T14:15:22Z", "endDate": "2019-08-24T14:15:22Z", "user": "../example", "machine": "../example" }]**
Response code	**200 (OK)**

API method	**Create history POST**
Purpose	**Create history entry**
Route	**/api/histories**
Request structure	**{ "startDate": "2019-08-24T14:15:22Z", "endDate": "2019-08-24T14:15:22Z", "user": "../example", "machine": "../example" }**
Header	**Authorization: Bearer {token}**

Response structure	{
	"id": 0,
	"startDate": "2019-08-24T14:15:22Z",
	"endDate": "2019-08-24T14:15:22Z",
	"user": "../example",
	"machine": "../example"
	}

Response code	201 (OK)

API method	Show history entry GET
Purpose	Show history entry
Route	/api/histories/{id}
Request structure	-
Header	Authorization: Bearer {token}

Response structure	{
	"id": 0,
	"startDate": "2019-08-24T14:15:22Z",
	"endDate": "2019-08-24T14:15:22Z",
	"user": "../example",
	"machine": "../example"
	}

Response code	200 (OK)

API method	Edit history entry PATCH
Purpose	To change history information
Route	/api/histories/{id}

Request structure	{
	"startDate": "2019-08-24T14:15:22Z",
	"endDate": "2019-08-24T14:15:22Z",
	"user": "../example",
	"machine": "../example"
	}

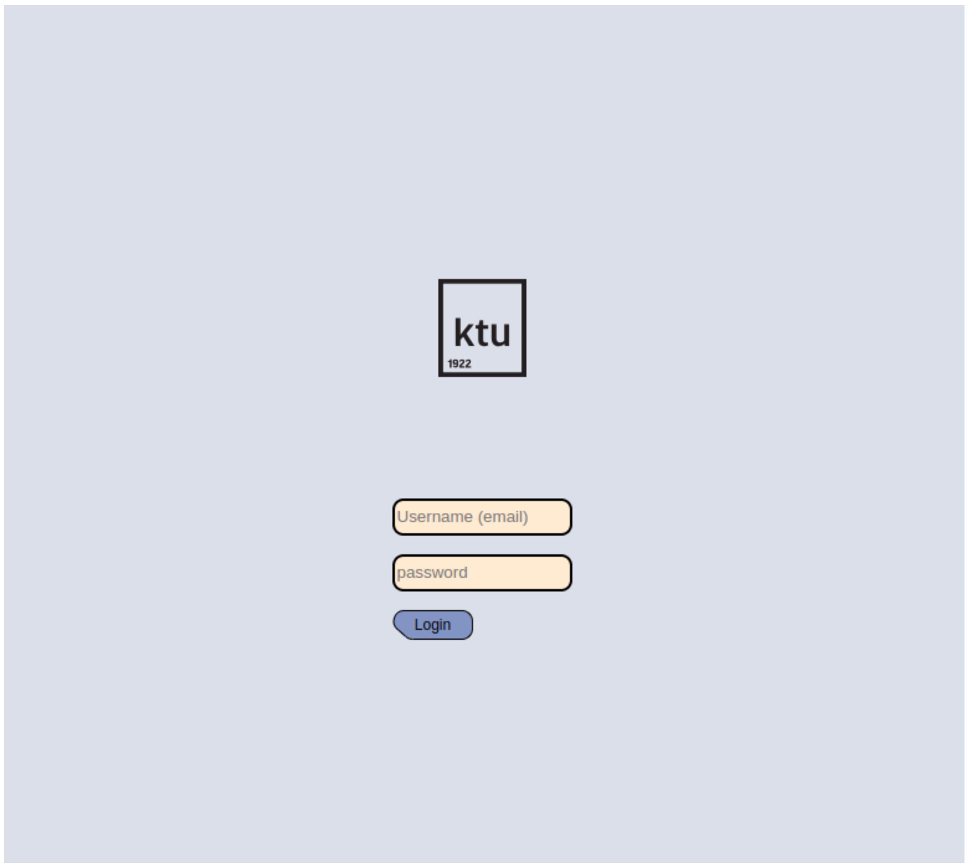
Header	Authorization: Bearer {token}

Response structure	**{ "id": 0, "startDate": "2019-08-24T14:15:22Z", "endDate": "2019-08-24T14:15:22Z", "user": "../example", "machine": "../example" }**
Response code	**200 (OK)**

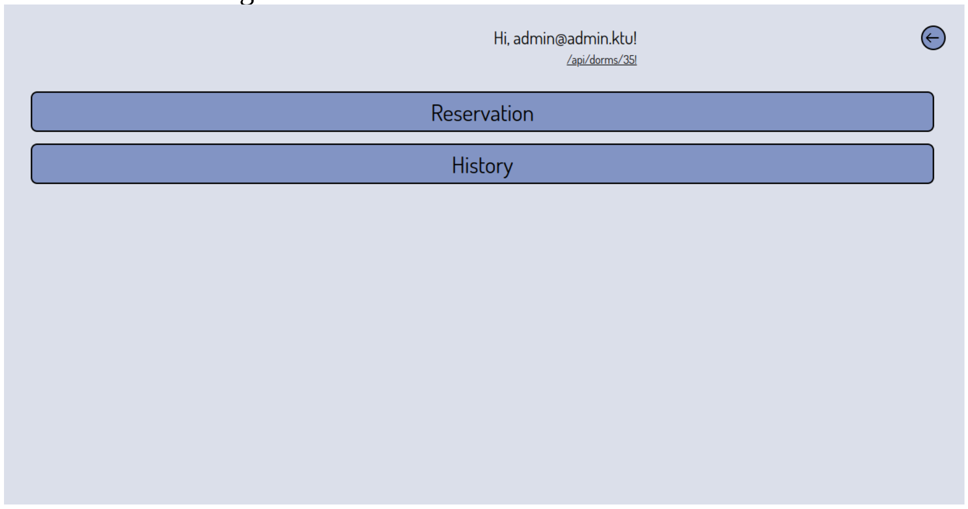
API method	**Remove history entry DELETE**
Purpose	**Delete history entry**
Route	**/api/histories/{id}**
Request structure	**-**
Header	**Authorization: Bearer {token}**
Response structure	**-**
Response code	**204 (OK)**

System user interface

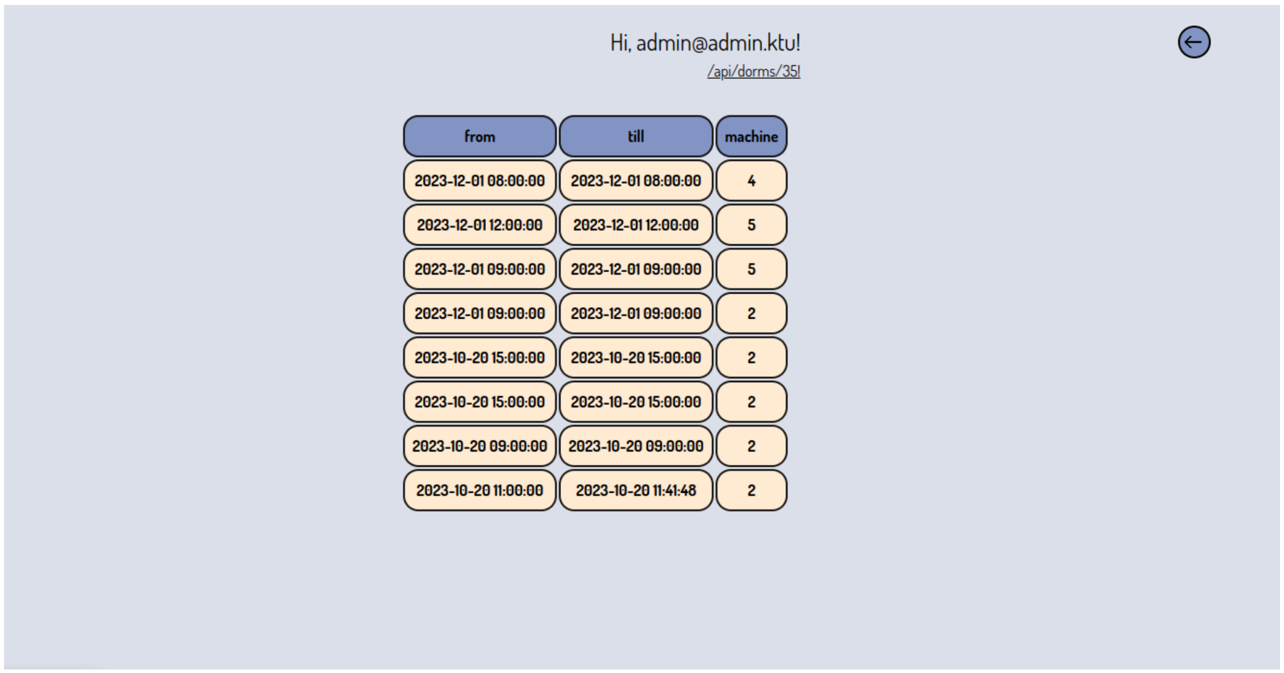
There are three user roles, user, dorm administrator and the administrator. Logged out users firstly see this login screen:



After successful login user sees this screen:



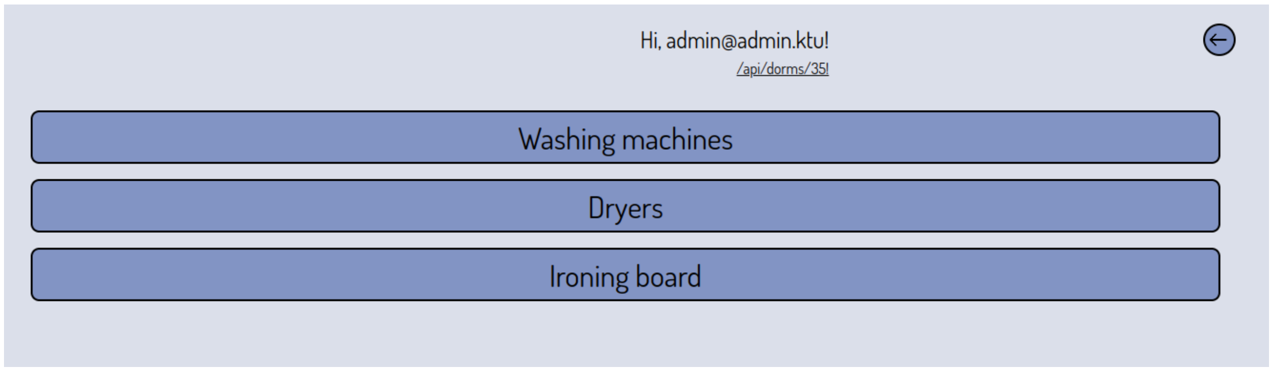
History screen:


Hi, admin@admin.ktu!

/api/dorms/35!

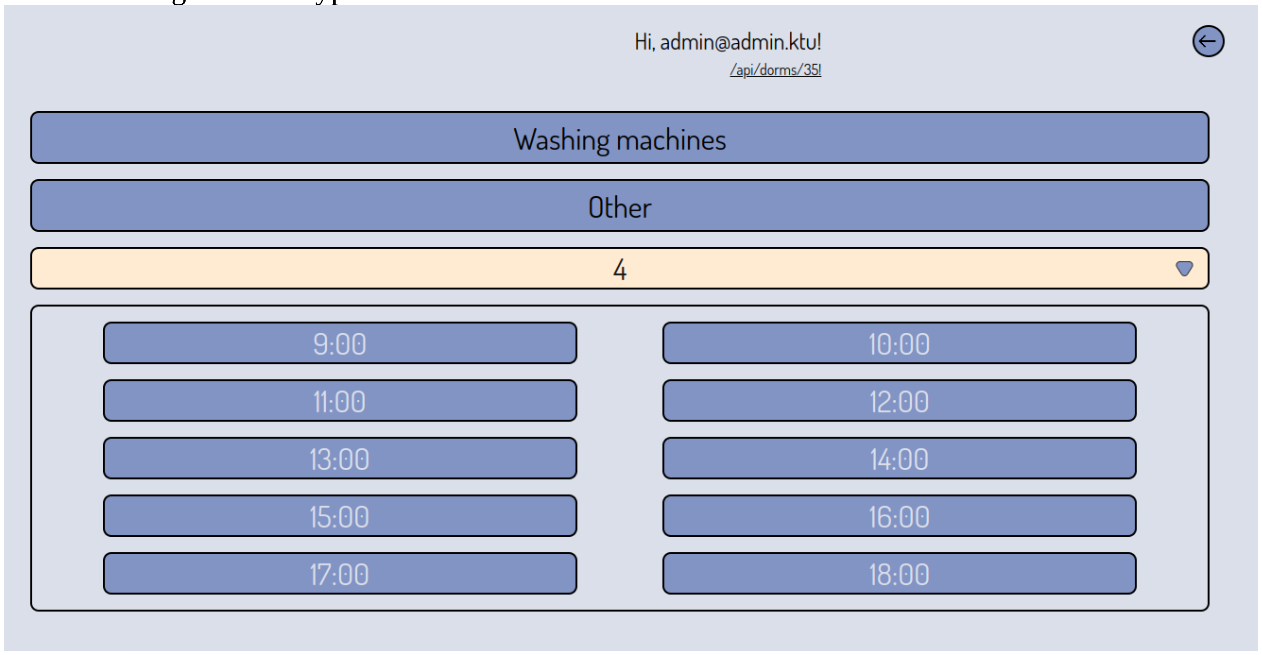
from	till	machine
2023-12-01 08:00:00	2023-12-01 08:00:00	4
2023-12-01 12:00:00	2023-12-01 12:00:00	5
2023-12-01 09:00:00	2023-12-01 09:00:00	5
2023-12-01 09:00:00	2023-12-01 09:00:00	2
2023-10-20 15:00:00	2023-10-20 15:00:00	2
2023-10-20 15:00:00	2023-10-20 15:00:00	2
2023-10-20 09:00:00	2023-10-20 09:00:00	2
2023-10-20 11:00:00	2023-10-20 11:41:48	2

Reservation screen:

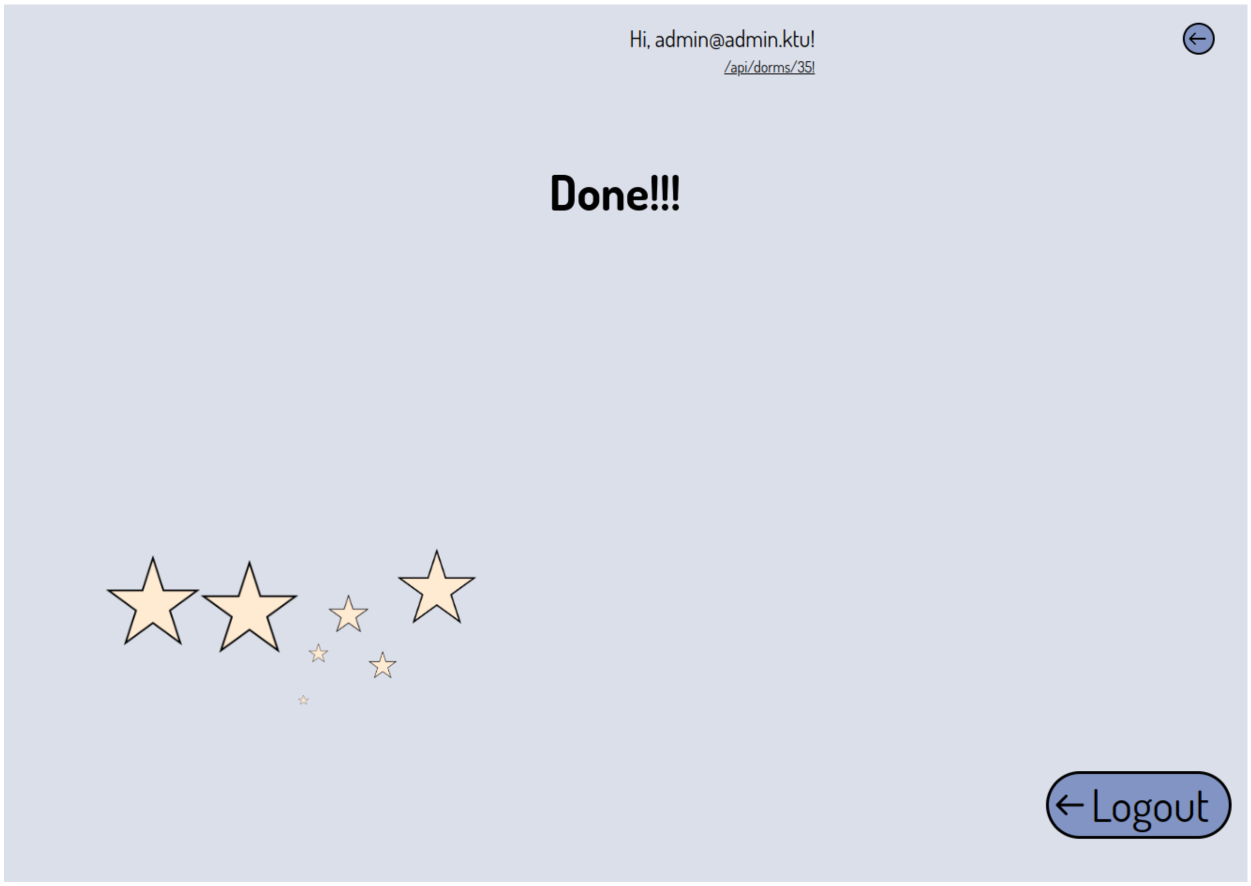

Hi, admin@admin.ktu!

/api/dorms/35!

After selecting machine type and number user sees:



Success screen:



Conclusions

- Implemented system to dorm residents to improve the quality of laundromat services
- The system consists of two parts: the client side and the server side
- Vue.js is used on the client side and PHP Symfony on the server side
- Implemented interface based on REST principles
- Implemented JWT based authentication / authorization
- A user interface was created, implementing REST API methods