Architecture Documentation

Rudolf Beck

Michaela Lodermeyer

Christian Dudas

Sven Weiler

Kenny Tran

SEW EURODRIVE GmbH
Bruchsal

30.06.2020

Table of content

1. Introduction and targets

1.1. Terms of reference

Package existing training information and content into a virtual game world. Creating a virtual PM world for learning, trying out and simulating realistic project decisions in a playful way. Integration of creative elements for motivation, documentation of learning progress and possible development paths (ranking, archievments, quest roadmap).

1.2. Targets

Construction of a modular game world with exemplary implementation of a module (kick-off). The application should contain motivational elements to keep the user's interest high. The achieved learning contents should be verifiable in the application. Instructions for future extensions must be provided.

1.3. Basics

The application must be browser-based and must be able to be used in the Internet Explorer browser. The user should be able to play via a laptop or tablet. It should be short game units in English language. No other hardware needs to be used for the application. The application is built up in modules and can be extended for further use.

1.4. Delimitation

Language basis for implementation and documentation is English. The extension to other languages is not given. The performance of the application comes before the graphical representation.

1.5. Target group

- Project managers with basic knowledge
- Beginners with first practical experience
- Internationally active
- Engineers
- Customer projects 4.0

2. Boundary conditions

The project is provided as a web application. The application shall be used mainly via the browser Internet Explorer. After completion of the implementation project for the kick-off module, the application shall continue to be used. New content should be easily extendable. Each module should be completed by one user in 15 minutes maximum. In order to enable use for all locations worldwide, the application will be designed in English. The application should illustrate the effects of decisions on the project. This, as well as other implementation ideas that should serve as motivation, can be freely chosen during implementation. For the color design, reference is made to the logo and color palette provided by SEW.

Technical boundary conditions are not given. As a result, the programming language, the framework and the database can be freely chosen. However, there are requirements for the database which have to be considered when choosing the product. In the following the requirements are listed:

- Preferably a Microsoft product
- User data for authentication should be persisted
- Data that serves the gamification aspect, such as rankings or achievements must be assigned to the user
- Since the "game" is extensible (by SEW as well as by the customer, depending on creativity), it can be assumed that, depending on the extension, additional information may need to be persisted
- Read and write access to user data (authentication) and data concerning gamification aspects (achievements, ranking, ...)

3. Context delimitation

No external interfaces to neighbouring systems are provided. To prevent the loss of information, a database is to be integrated. The Web application does not differentiate between different user roles and authorizations.

4. Solution strategy

4.1. Technology decisions

4.1.1. Database

It must be assumed that new types of information must be assigned to users depending on the creativity of the developer. NoSQL databases are better suited for such a use case than conservative relational databases. With fewer "joins" NoSQL provides faster read access (performance +) at the expense of consistency. NoSQL databases can be roughly divided into the following types:

- Document Store
- Key-Value Store
- Wide-column Store
- Graph database

Graph databases and key-value stores are dropped because there is neither data for graphs (nodes and edges) nor value-key pairs. Document stores, on the other hand, are ideal for frequently changing requirements.

Conclusion: Document stores are recommended for the application to be built. Since SEW already has a Mongo DB in use, it was decided to use it. This decision is also supported by the positive experience of the developers in this project.

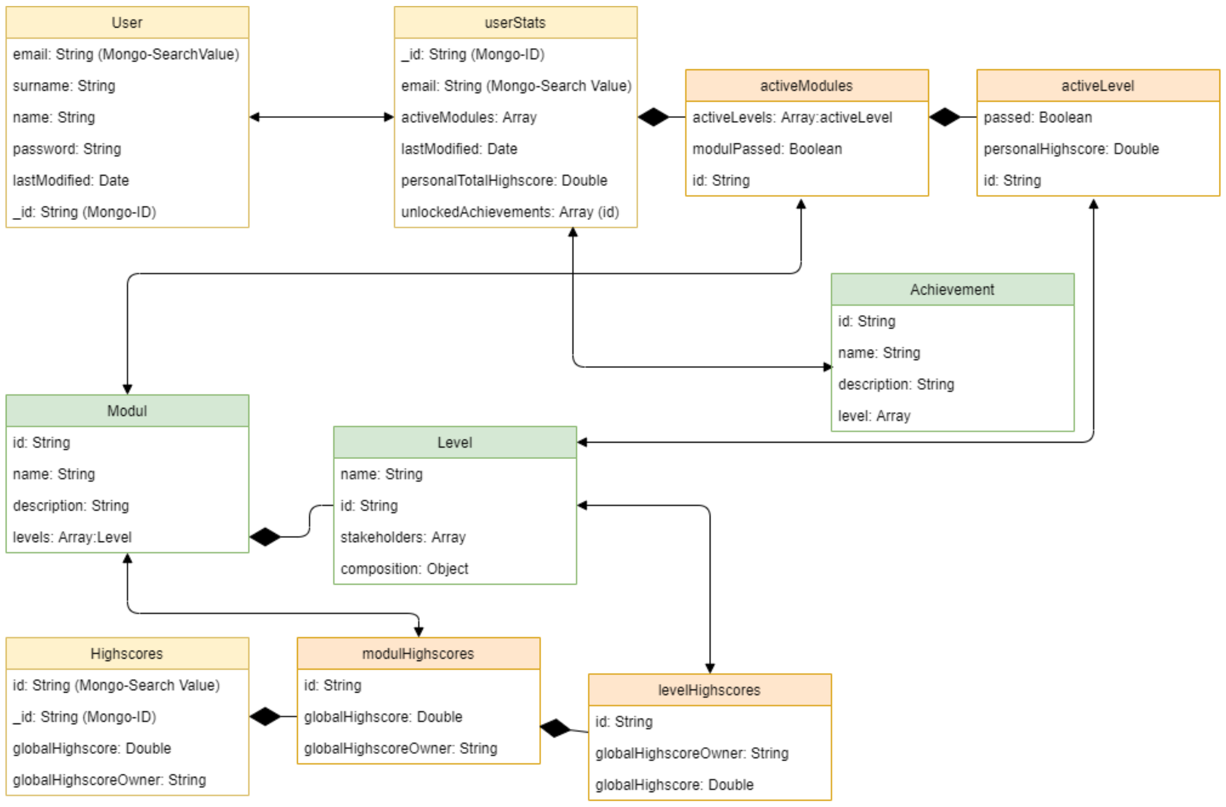
4.1.2. Backend / API:

Express is used as the solution to be implemented. It is well suited for both small and large projects. It has a large community with about 6.3 million users according to github. Express is a framework on the market with the standard selection node.js which has been matured for 10 years. A node.js based framework is preferred, because then only one programming language (Typescript/ Javascript) has to be developed

4.1.3. Frontend:

The development of the frontend is supported by the web application framework Angular.

4.2. Object disassembly



4.3. Quality requirements

The quality of the API is ensured by automated tests. The quality of the front-end is ensured by manually logged click tests.

5. Module view

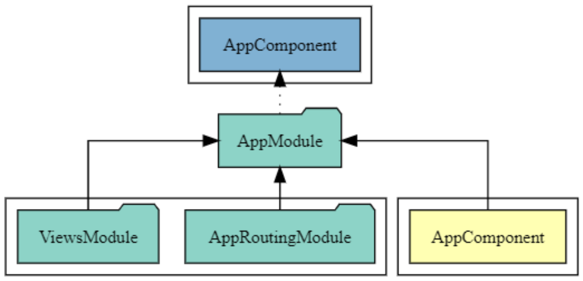
5.1. Modules and Components

Legend for the following diagrams:

🟨 Declarations 🟩 Module 🟥 Exports

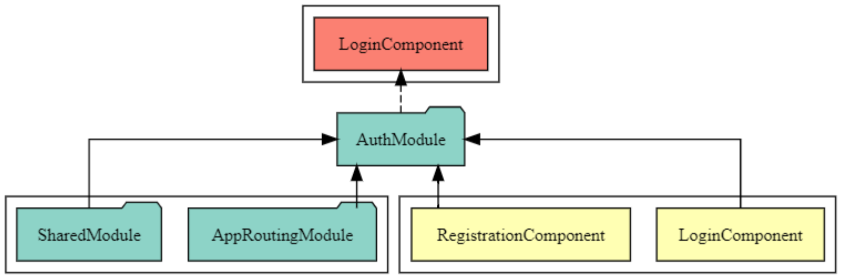
➢ AppModule

Declarations	Imports	Exports	Bootstrap
AppComponent	AppRoutingModule	ViewsModule	AppComponent

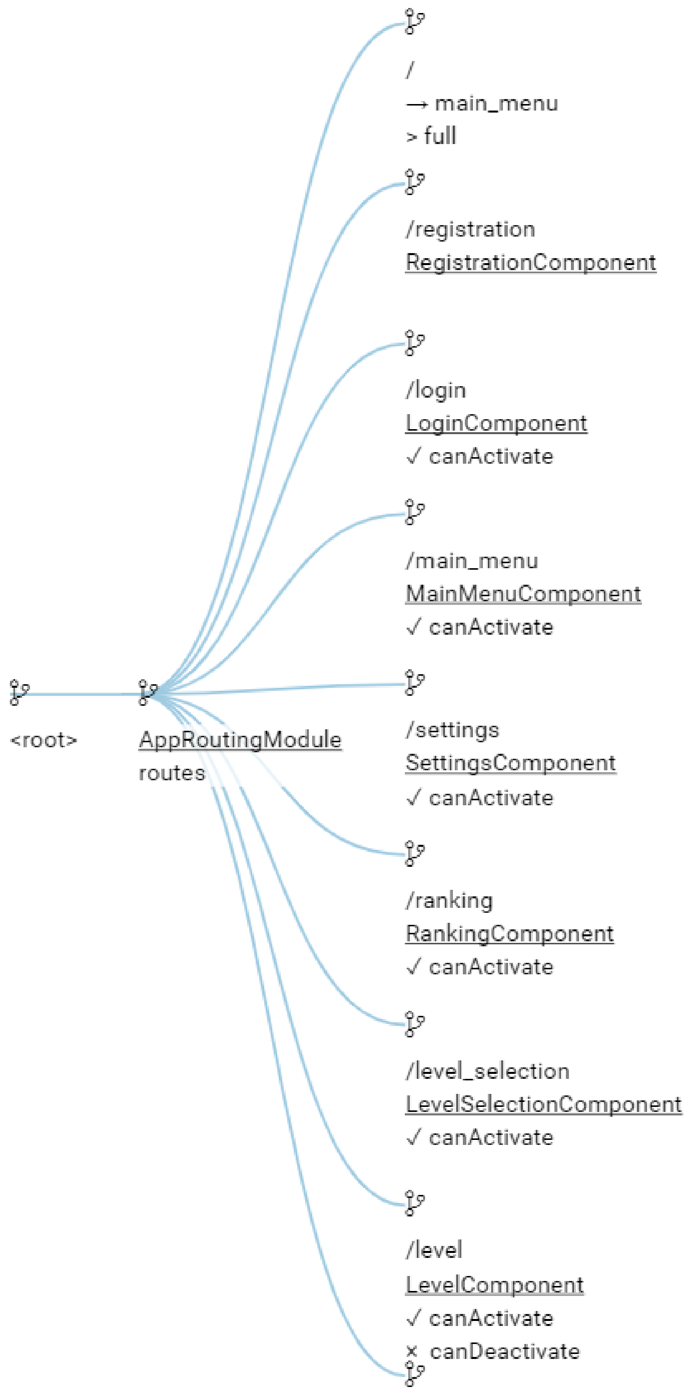


➢ AuthModule

Declarations	Imports	Exports
LoginComponent	AppRoutingModule	LoginComponent
RegistrationComponent	SharedModule	

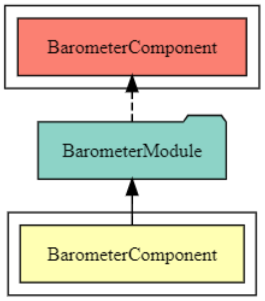


- AppRoutingModule



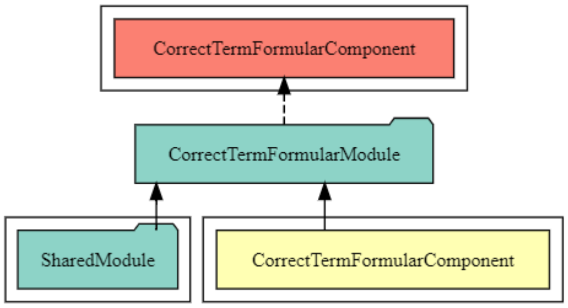
- BarometerModule

Declarations	Imports	Exports
BarometerComponent		Barometercomponent
RegistrationComponent		



- CorrectTermFormularModule

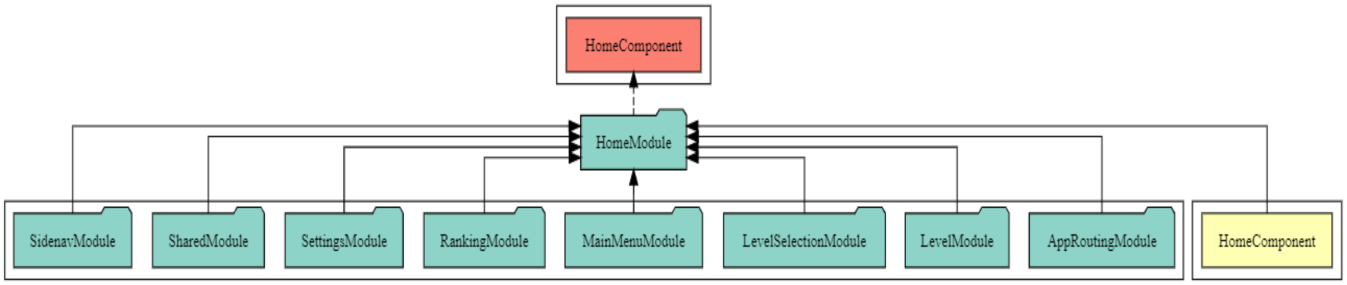
Declarations	Imports	Exports
CorrectTermFormularComponent	SharedModule	CorrectTermFormularComponent


CorrectTermFormularComponent
CorrectTermFormularModule
SharedModule
CorrectTermFormularComponent

- HomeModule

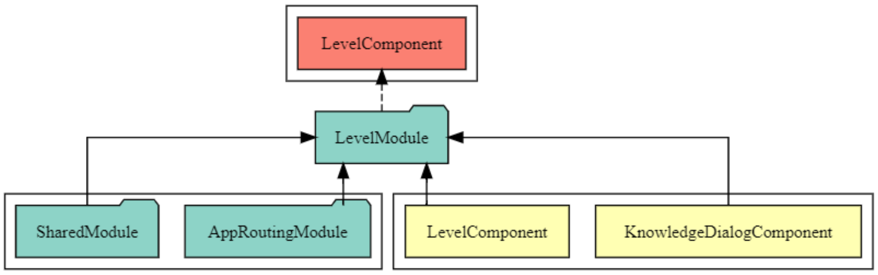
Declarations	Imports	Exports
HomeComponent	AppRoutingModule	HomeComponent
	LevelModule	
	LevelSelectionModule	
	MainMenuModule	
	RankingModule	
	SettingsModule	

	SharedModule	
	SidenavModule	



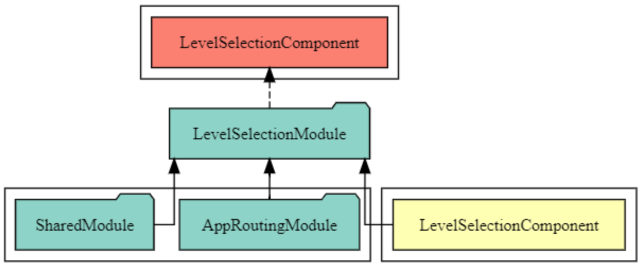
- LevelModule

Declarations	Imports	Exports
KnowledgeDialogComponent	AppRoutingModule	LevelComponent
LevelComponent	SharedModule	



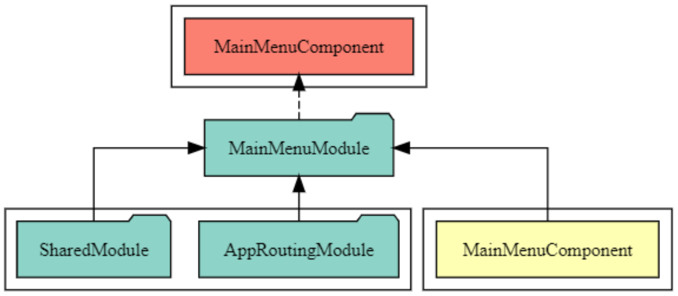
- LevelSelectionModule

Declarations	Imports	Exports
LevelSelectionComponent	AppRoutingModule	LevelSelectionComponent
	SharedModule	



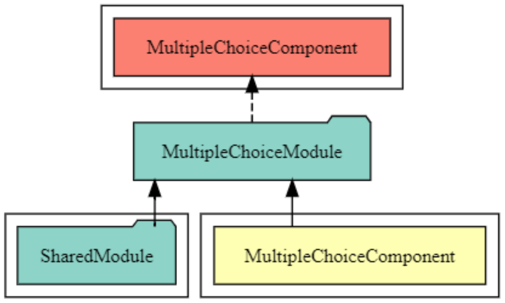
- MainMenuModule

Declarations	Imports	Exports
MainMenuComponent	AppRoutingModule	MainMenuComponent
	SharedModule	



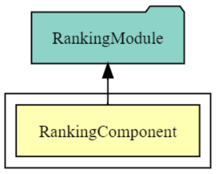
- MultipleChoiceModule

Declarations	Imports	Exports
MultipleChoiceComponent	SharedModule	MultipleChoiceComponent



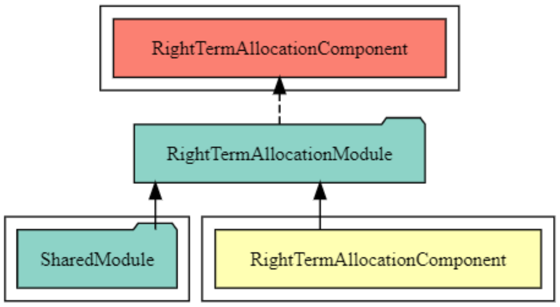
- RankingModule

Declarations	Imports	Exports
RankingComponent		



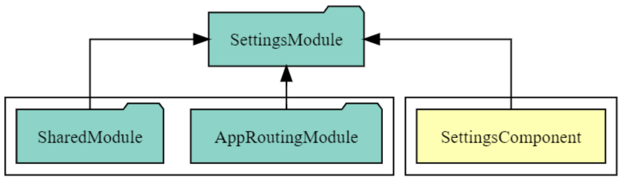
- RightTermAllocationModule

Declarations	Imports	Exports
RightTermAllocationComponent	SharedModule	RightTermAllocationComponent


RightTermAllocationComponent
RightTermAllocationModule
SharedModule
RightTermAllocationComponent

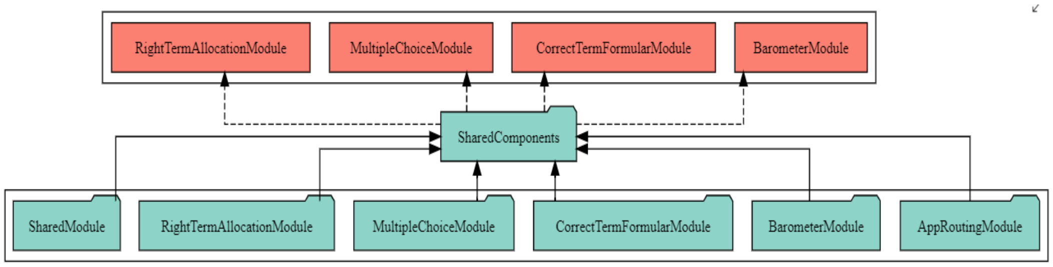
- SettingsModule

Declarations	Imports	Exports
SettingsComponent	AppRoutingModule	
	SharedModule	



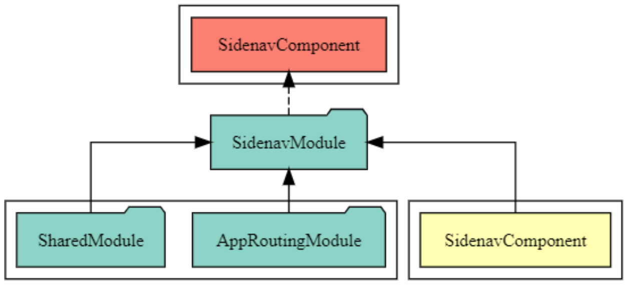
- SharedModule

Declarations	Imports	Exports
	AppRoutingModule	BarometerModule
	BarometerModule	CorrectTermFormularModule
	CorrectTermFormularModule	MultipleChoiceModule
	MultipleChoiceModule	RightTermAllocationModule
	RightTermAllocationModule	



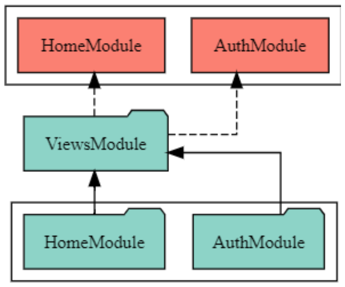
- SidenavModule

Declarations	Imports	Exports
SidenavComponent	AppRoutingModule	SidenavComponent
	SharedModule	



- ViewsModule

Declarations	Imports	Exports
	AuthModule	AuthModule
	HomeModule	HomeModule



5.2. Dependencies

Dependency name	Dependency version
@angular/animations	9.1.1
@angular/cdk	9.2.1
@angular/common	9.1.1
@angular/compiler	9.1.1
@angular/core	9.1.1
@angular/forms	9.1.1
@angular/material	9.1.1
@angular/platform-browser	9.1.1
@angular/platform-browser-dynamic	9.1.1
@angular/router	9.1.1
chart.js	2.9.3
chartjs-plugin-datalabels	0.7.0
d3-interpolate	1.4.0
ng2-charts	2.3.2
ngx-animate	1.0.1
ngx-cookie-service	3.0.4
rxjs	6.5.4
tslib	1.10.0
zone.js	0.10.2
lodash	4.17.15

5.3. Frameworks

Frontend:

Angular 9 (latest stable version: 9.1.0 / 25 March 2020)

API:

Express – Node.js

5.4. Three-Tier-Modell

For a better understanding of the project, the code is "physically" separated to provide insight into how the entire project is structured. The Three-Tier-Modell is used for this purpose.

Presentation Layer: The frontend of the application using the web framework Angular.

Application Layer: This layer is the “engine” of the project and it connect the presentation layer to the data layer. Implemented in JavaScript and using Express JS.

Data Layer: The database using MongoDB. The data is accessed from the application layer via API calls.

6. Design decision

See design specification document.