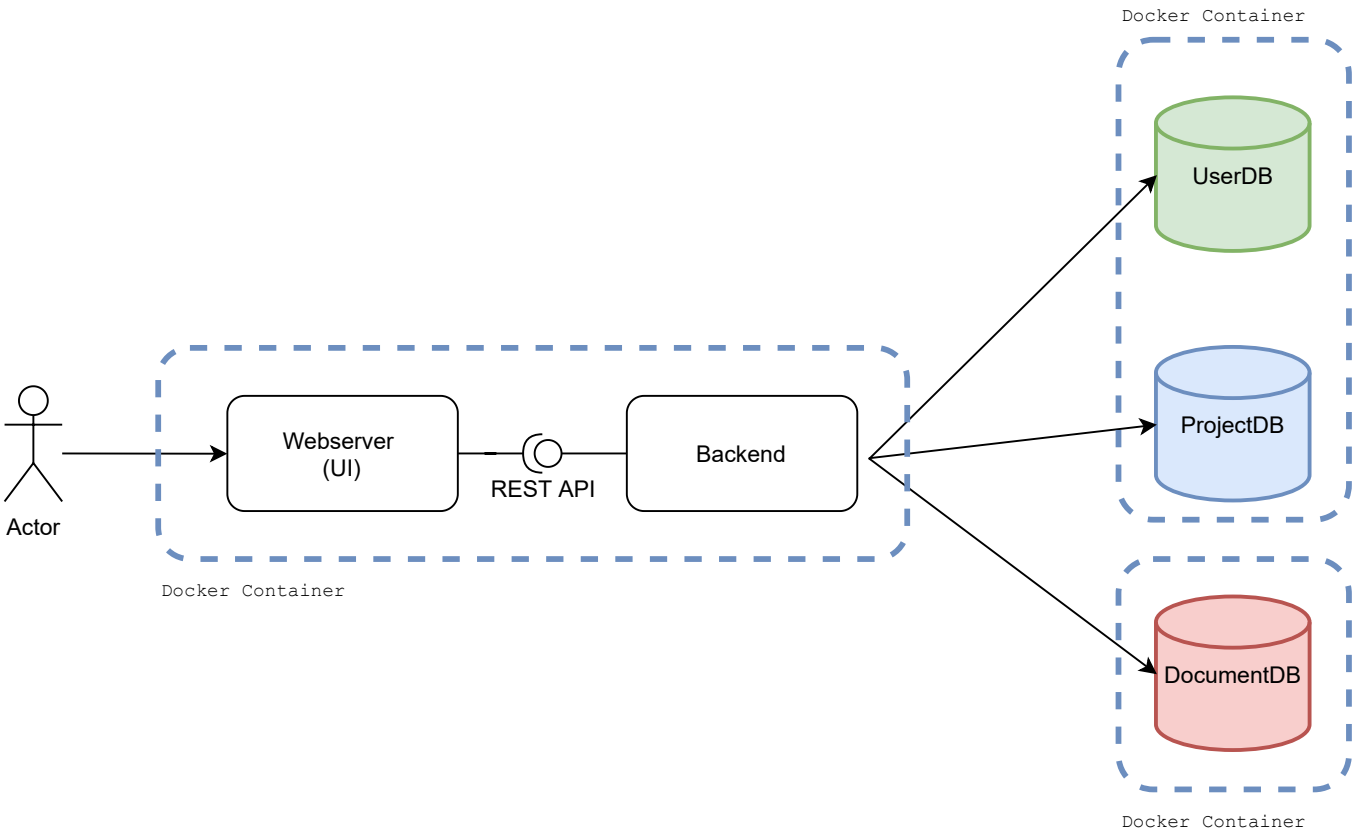
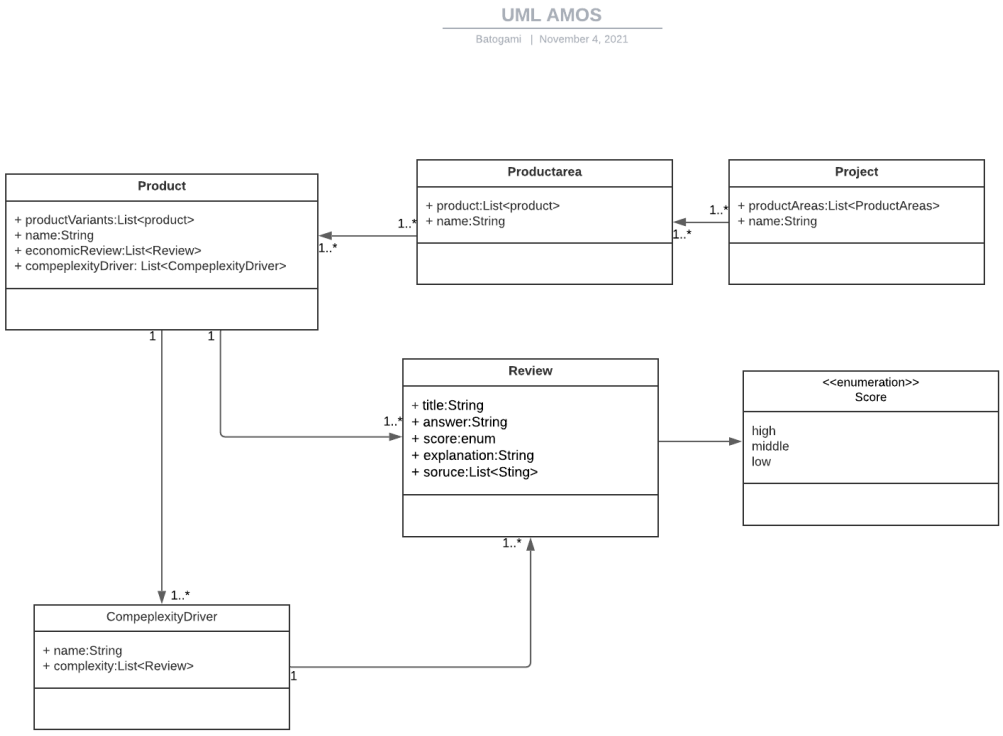
Architecture: runtime components

Version: 1.0



Explanation: code components



This diagram shows the class structure of the part of our project which is independent from frameworks or similar things (like the different controllers required by Spring Boot for the REST API).

Each project has a name and a specific number of product areas . In the current version a project can have three different product areas kredit, payment, and customer. Since the number of product areas can increase it’s shown as a one-to-many relationship. A project must have at least one product area.

A product area has a name and at least one product. A product can have variants and consists further of name and economic and complexity reviews. The complexity reviews are ordered by complexity drivers. A review consists of a title (a name or question), an answer, the score given by a user and an explanation for the score as well as a list of sources.

	Area	Infrastructure		
	Tech	**Summary**	**Version**	**License**
	Github Actions	CI/CD Tool for running automated tests, building docker images and deployment.	-	-
	Docker Engine	Docker engine is used to run applications in a container	v20.10	Apache License 2.0
	Docker CLI	Tool to build containerized applications as docker images.	v20.10	Apache License 2.0
	Area	Backend		
	Tech	**Summary**	**Version**	**License**
	Spring Boot	Used for REST API and microservice architecture	2.5.6	Apache License, Version 2.0
	HyperSQL	Deployment of database applications	2.6.2	BSD License
	Area	Frontend		
	Tech	**Summary**	**Version**	**License**
	React	UI Framework and handler for the view layer of web and mobile apps	17.0.1	MIT License

Our software is a RESTful web service. The service provides the complexity evaluation of financial product portfolios as well as the corresponding data visualizations and user interface. Therefore, it can be divided into three major parts, fronted as a web UI, backend, and the databases.

To enable RESTful communication between all parts, we are using the Spring Boot framework. The controller classes provided by this framework will be used to create the REST API while the framework itself can provide a web UI itself from the necessary HTML files. For large amounts of data/variables JSON files will be used for the communication to prevent an overload of the controller parameters and to ease the conversion to classes, database entries, or a class to a UI visualization. Also Spring Boot provides inbuilt functions for login and user management.

For the project and user management a database system will be used. At the moment we prefer HSQLDB (HyperSQL Datenbank). A relational database system written in Java (http://hsqldb.org/).

The content database should hold all information about the projects. The user database will be used together with the Spring Boot framework to restrict the access to our software and to distinguish between the roles of users e.g., admin, user, project manager.

For the deployment docker will be used. Our microservices will be packed into containers which will allow for easier scaling by simply adding more containers on demand. Testing will be done by pushing microservice applications into a test environment and executing both automated and manual tests.