aMAZEing

Badarau Bogdan, Olariu Leonard, Popa Dorian

Context

This project idea came after watching Clement Mihailescu's video, where he presented many different projects he worked on over time and helped him get a job at companies like Google. Considering our competitive programming background, one specific idea stood out for us. It was basically a web visualizer for path finding algorithms. We really wanted to do something similar, because it is as much user-oriented, as it is developer oriented.

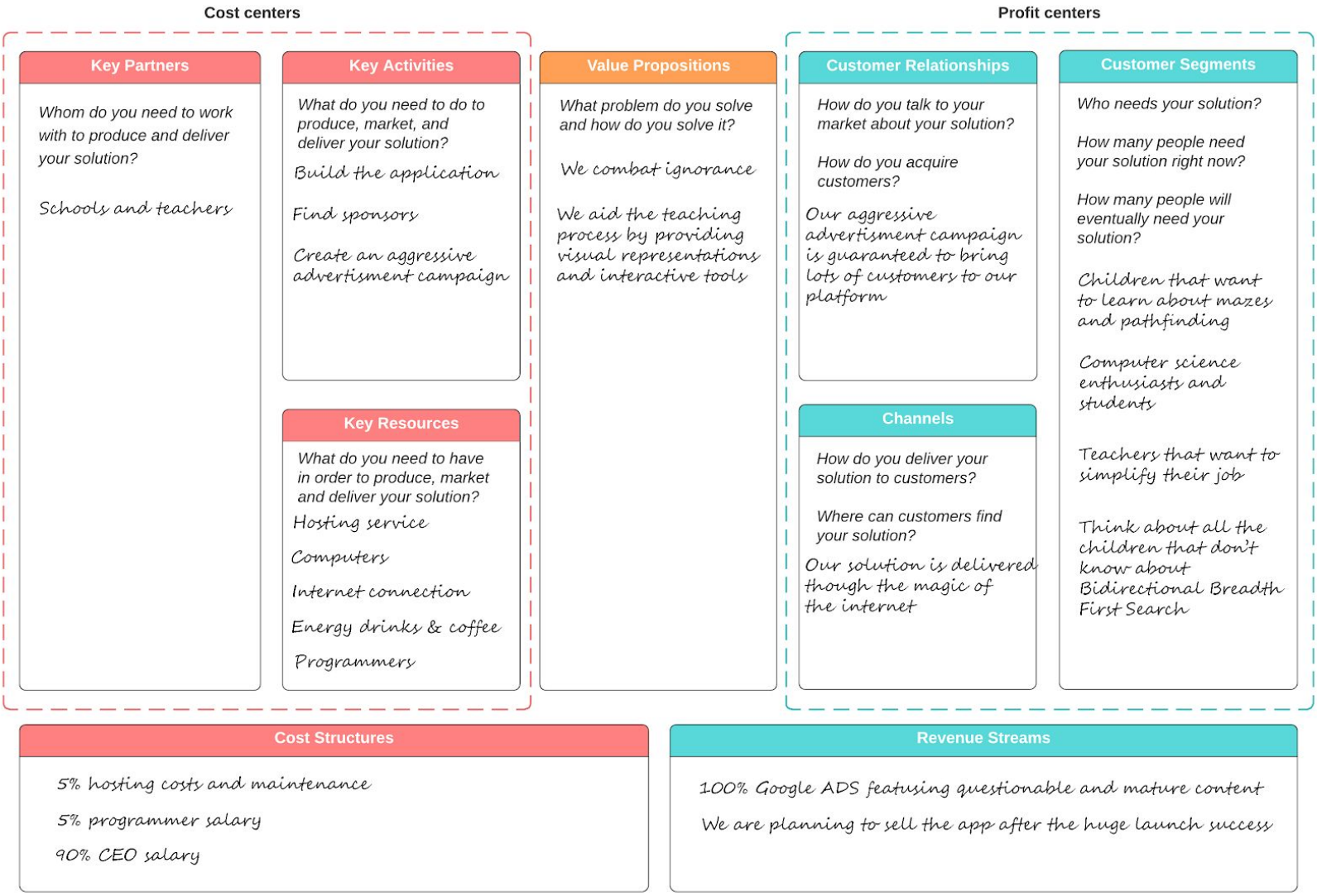
User-oriented:
- being very visual and easy to use
- having educative purpose

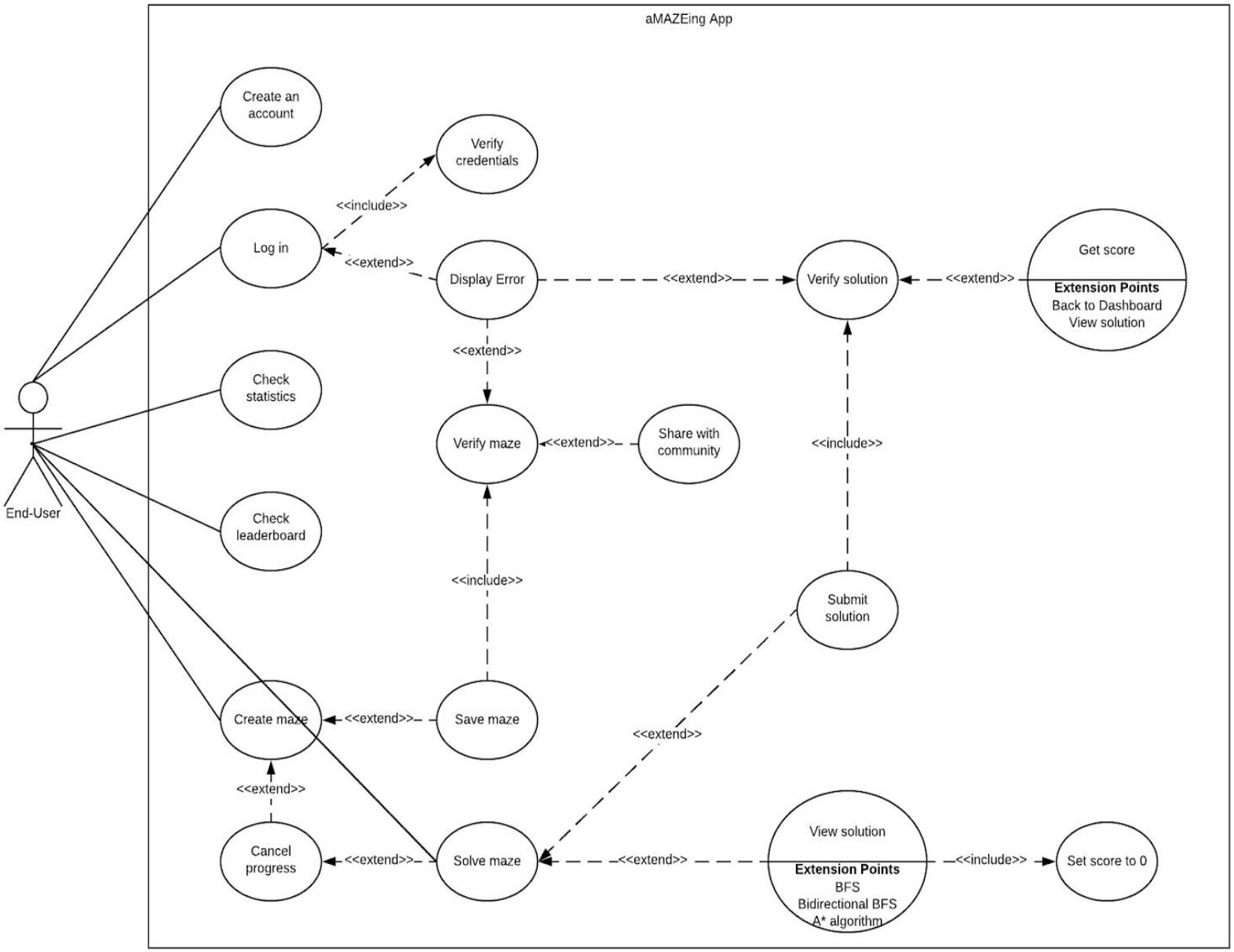
Developer-oriented:
- being challenging to develop
- offering a great learning experience
- representing a good practice for code interviews

As inspiration we used these two applications that developed this idea: Clement's and Xueqiao's.

Business model

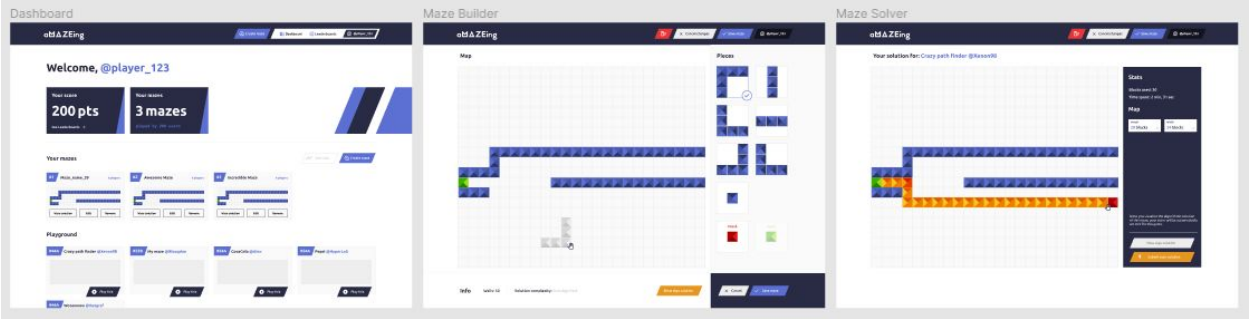
Analyzing the original idea that focused on aiding the teaching process, we felt that it might benefit from adding a human interaction component. The way we achieved that is by letting people share mazes that they created with the community. Solving a maze from the playground gives an accuracy rating to the user. Based on everyone's average accuracy we create a leaderboard. This motivates people to take their time and think about the maze they are trying to solve.




aMAZEing App
Create an account
Verify credentials
<<include>>
Log in
<<extend>>
Display Error
<<extend>>
Verify solution
<<extend>>
Get score
Extension Points
Back to Dashboard
View solution
<<extend>>
Check statistics
Verify maze
<<extend>>
Share with community
<<include>>
End-User
Check leaderboard
<<include>>
Submit solution
Create maze
<<extend>>
Save maze
<<extend>>
<<extend>>
Cancel progress
<<extend>>
Solve maze
<<extend>>
View solution
Extension Points
BFS
Bidirectional BFS
A* algorithm
<<include>>
Set score to 0

Tool stack

We used Figma to create the mock-ups for the app. The design had to be modern and intuitive.



The backend stack was based on .NET Core, Entity Framework and Microsoft SQL Server, while the frontend was developed using React, Redux and Redux-Saga. After integrating all our work we wanted to go online and made use of Cloud Services like Azure and Netlify to do that.

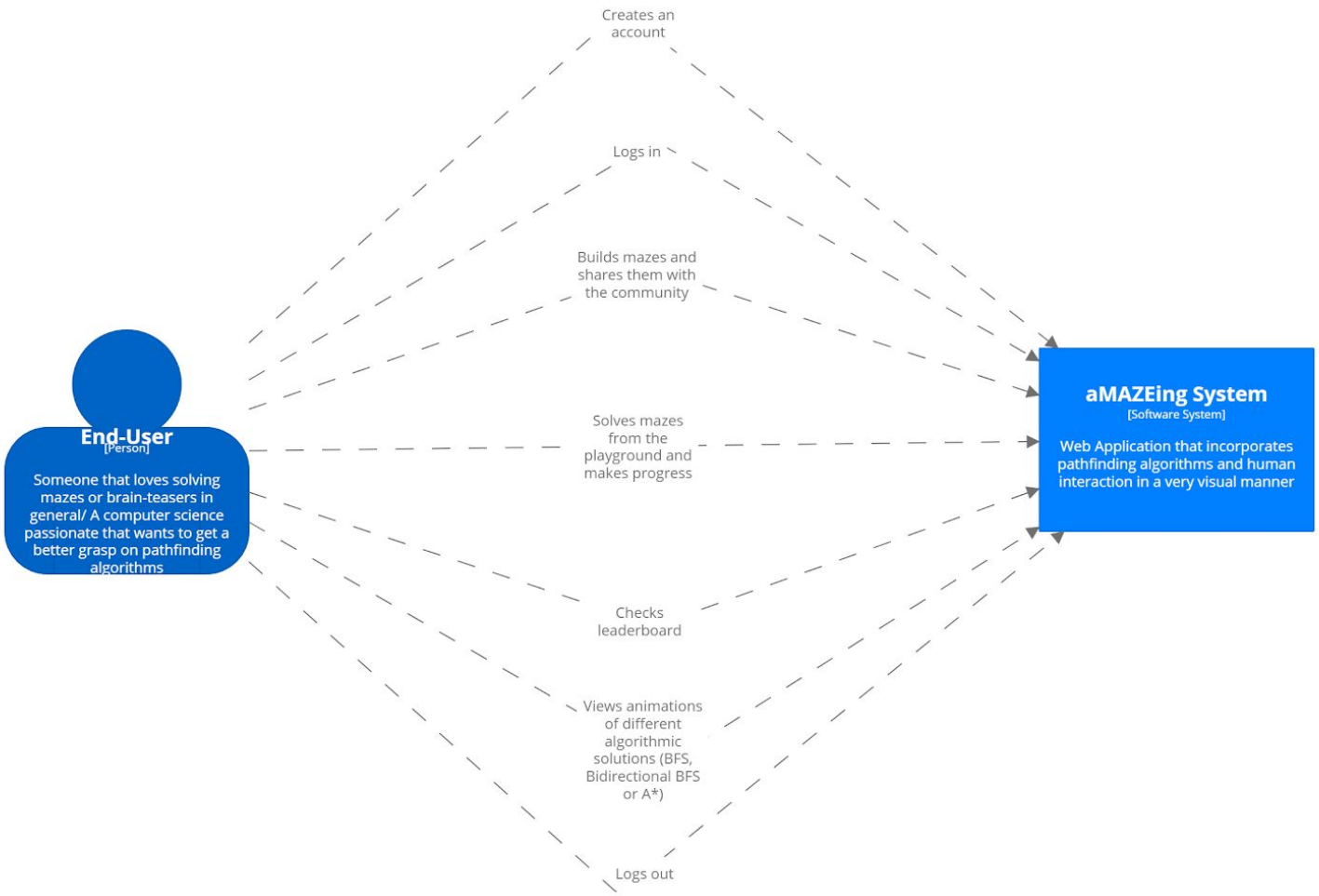
Development approaches

For our development approach we had to take into consideration that beside time spent in classes everyone was also working on personal projects. We concluded that definition driven development would be the best approach. We started defining our models and endpoints that threw not implemented exception. This way we could share a postman documentation with all the endpoints exposed by our API and ensure that all of us are on the same page.

It proved to be really helpful in aligning the frontend redux store with the backend models and, also really important to us, in decoupling our work! To reinforce this idea of decoupled work, we also setup a github repository.

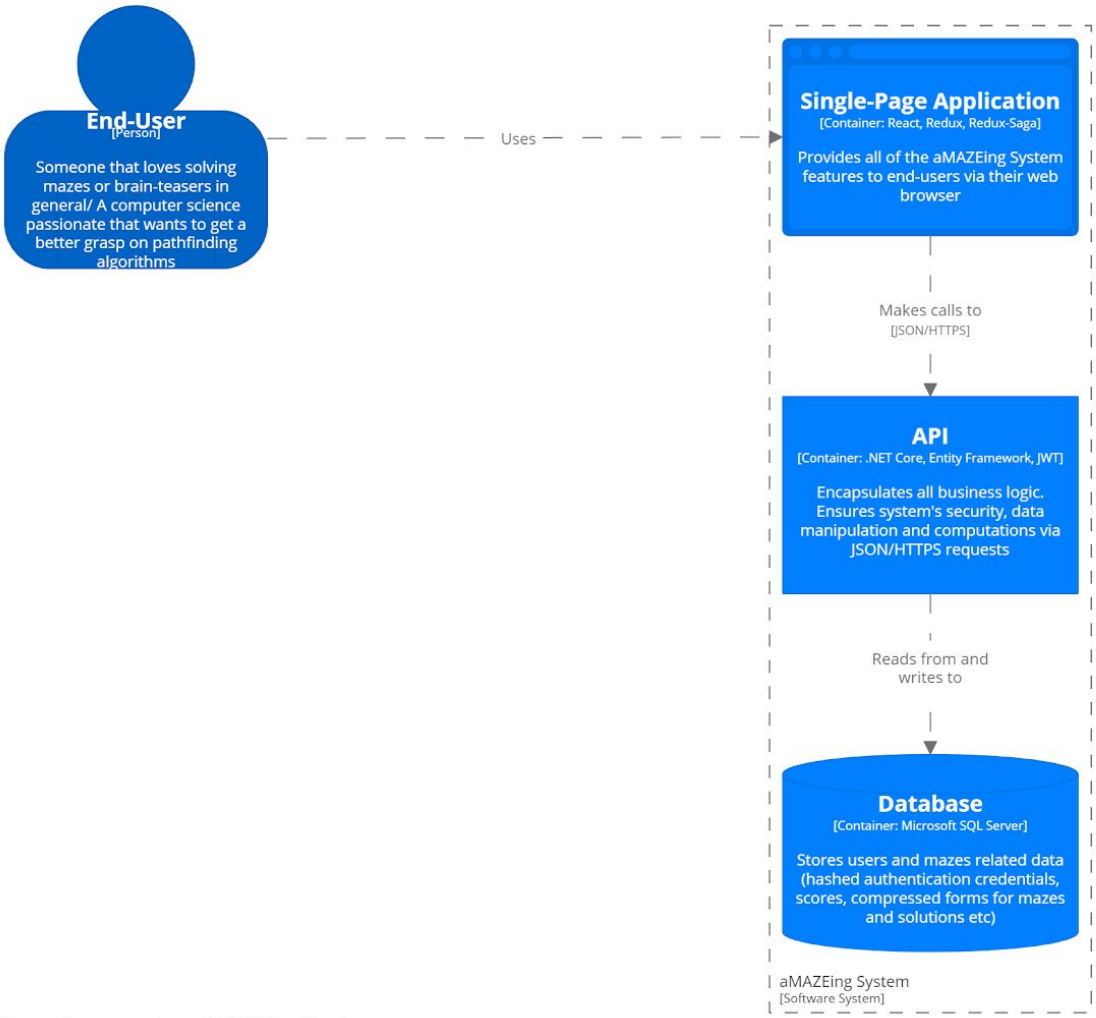
Moreover, there were many unclear things at the beginning, so we wanted to be as flexible as possible with data retrieved by our backend and we managed that by designing DTOs that were implementing builder design pattern. All these little things helped us a lot in delivering everything in time.

Architecture



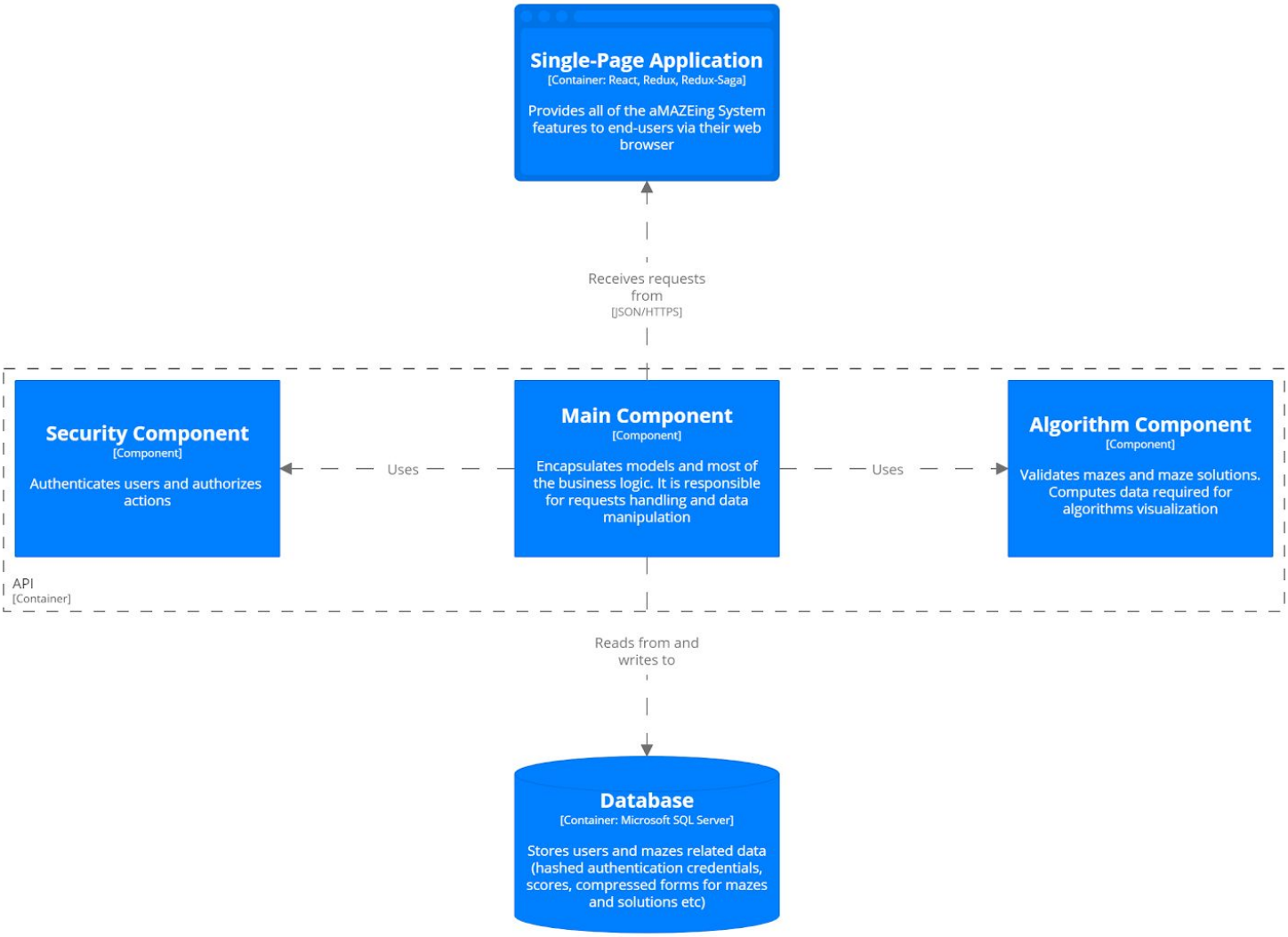
System Context diagram for aMAZEing System

Workspace last modified: Sun May 03 2020 15:26:57 GMT+0300 (Eastern European Summer Time)



Container diagram for aMAZEing System

Workspace last modified: Sun May 03 2020 17:42:46 GMT+0300 (Eastern European Summer Time)



Component diagram for aMAZEing System - API

Workspace last modified: Sun May 03 2020 18:13:59 GMT+0300 (Eastern European Summer Time)