Quantified Student

Software Architecture



Created by G. Malisz

Spis treści

[1. Version History 2](#_Toc131088204)

[2. Overview 2](#_Toc131088205)

[3. Microservices Architecture 2](#_Toc131088206)

[4. DDD (Domain Driven Design) 3](#_Toc131088207)

[5. Level 1: System Context diagram 5](#_Toc131088208)

[6. Level 2: Container diagram 6](#_Toc131088209)

[7. Level 3: Component diagram 7](#_Toc131088210)

[8. Level 4: UML 8](#_Toc131088211)

[9. Architecture of Microservices 9](#_Toc131088212)

[9.1. Microservice 1 9](#_Toc131088213)

[9.2. Microservice 2 9](#_Toc131088214)

[10. Definitions, Acronyms, and Abbreviations 10](#_Toc131088215)

# Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comment |
| 0.1 | 14-03-2023 | G. Malisz | First iteration of this document, some architecture components will have to be updated |
| 0.2 | 13-04-2023 | M. Raczkowski | Add information about DDD architecture. |
| 0.3 | 12-04-2023 | G. Malisz | Rework of the C4 model, layers 2 and 3 were remade with the compliance to the c4 model rules, added UML’s |
| 0.4 | 19-04-2023 | G. Malisz | Added information about the role of the gateway. |

# Overview

During the stand-ups and meetings of the team, there needed to be more clarity about the

responsibilities of each part of the system. The goal of this document is to create a clear and

concise overview of the Quantified Student system. This document functions as compendium of Quantified Student system, where every part of the system is presented in abstract way with usage of C4 model.

# Microservices Architecture

QS group has decided on Microservices Architecture, because:

* Quantified Student System in its final form is supposed to aggregate data sources, which provide useful information about students. This feature combined with analysis this data in future can cause some performance issues. Microservices architecture can help solve those problem by dividing application to smaller components that can be updated and scaled separately, without affecting availability of the whole application.
* Quantified Student System also consists of older sub-projects and incorporating them into one system can be done easier with usage of microservices architecture. This architecture allows for combining different technologies, which is useful while working with projects that are built independently.
* Microservices architecture will also allow group members to improve their knowledge in the field of software architectures, by allowing them to learn and use new architecture.

# DDD (Domain Driven Design)

Our main architecture is microservice architecture which is serviced-oriented architecture structural styler and arranges applications as a collection of loosely coupled, fine-grained services, communicating lightweight protocols. In other words, the whole application is made from smaller applications (services) that are responsible for certain parts and tasks of the software. For a single service, we decided to use DDD (Domain Driven Design) which is a design cantered around the domain, or sphere of knowledge, depending on the area of use, and solves complex domain models, connecting to the core business concepts. A domain refers to the characteristics of the problem which the proposed software is to solve. This fits our need for simplicity and focuses on single responsibility, needed in microservice services. Beneath the schematic of using DDD in microservices

Obraz zawierający diagram

Opis wygenerowany automatycznie

Figure 1- https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/

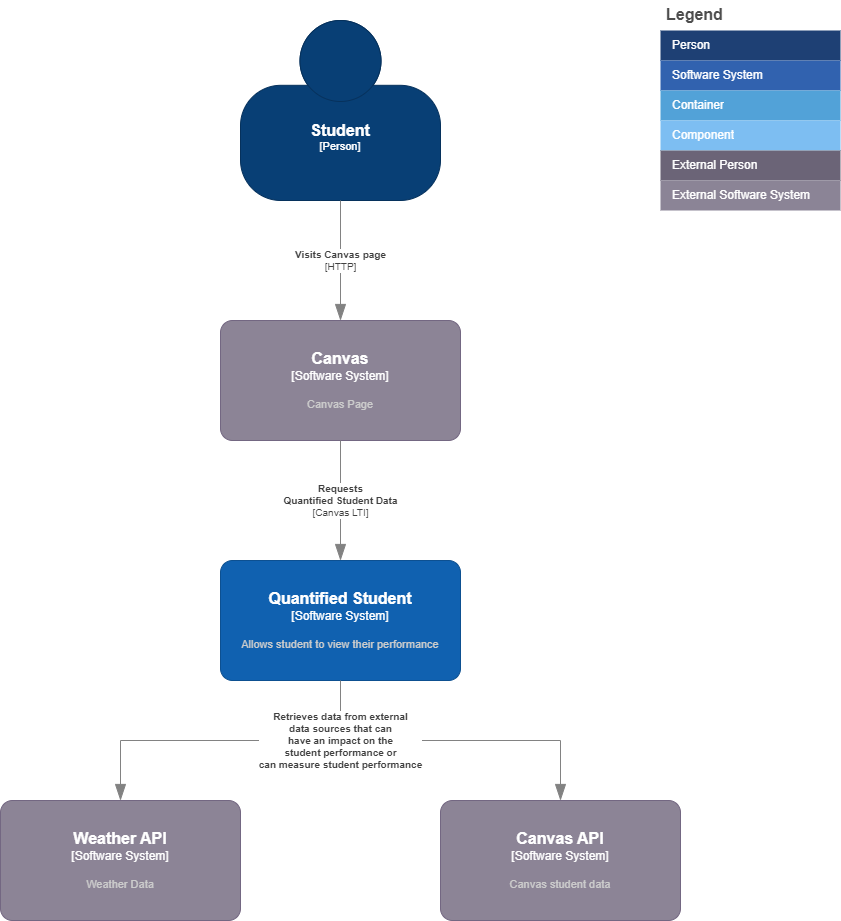
The layered nature of domain-driven design is also easy for maintainability extendibility and management, but most importantly it's a clear standard that can be followed by future colleges working on the project. Beneath the schematic of the DDD structure

Obraz zawierający diagram

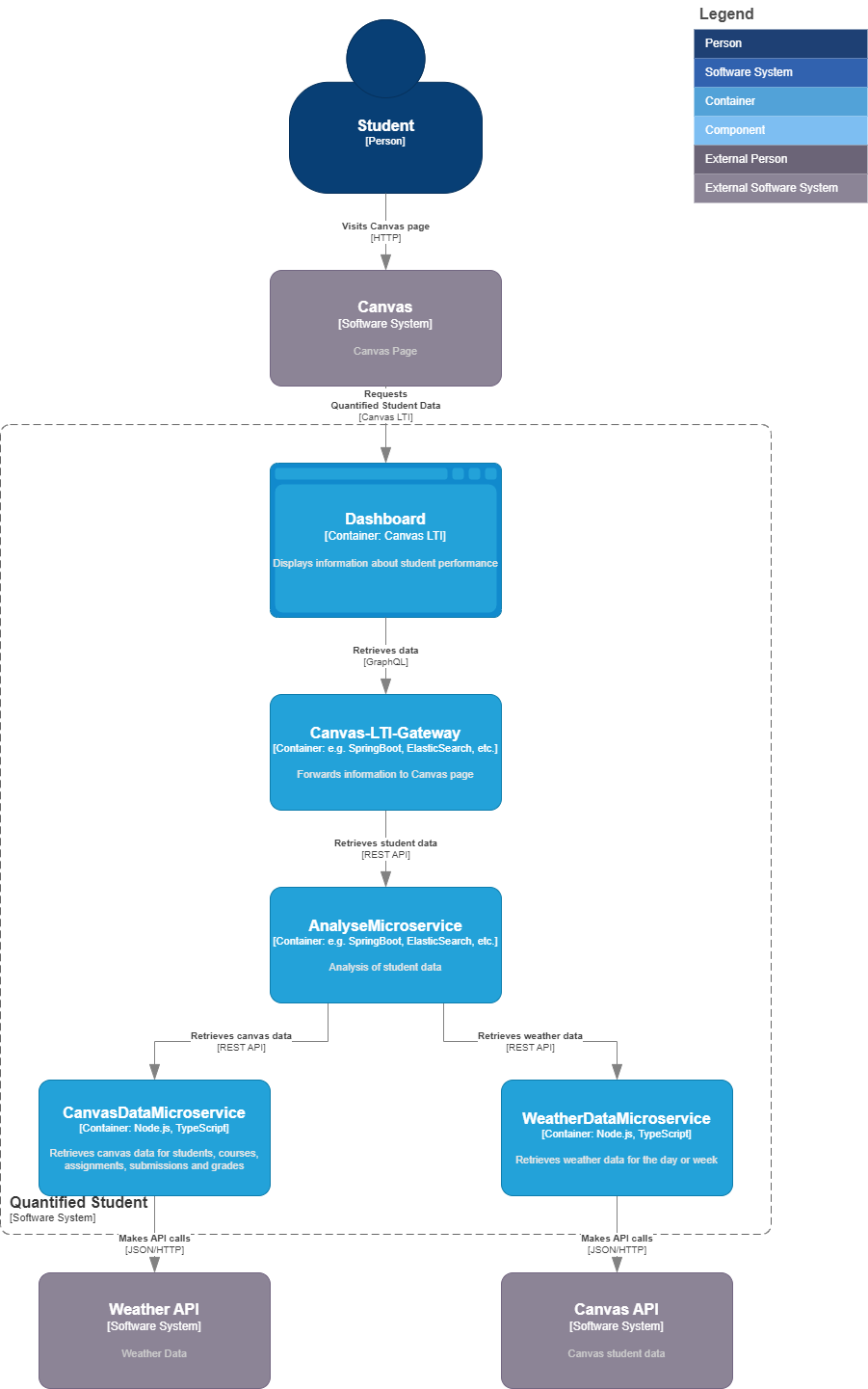
Opis wygenerowany automatycznie

Figure 2- https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice

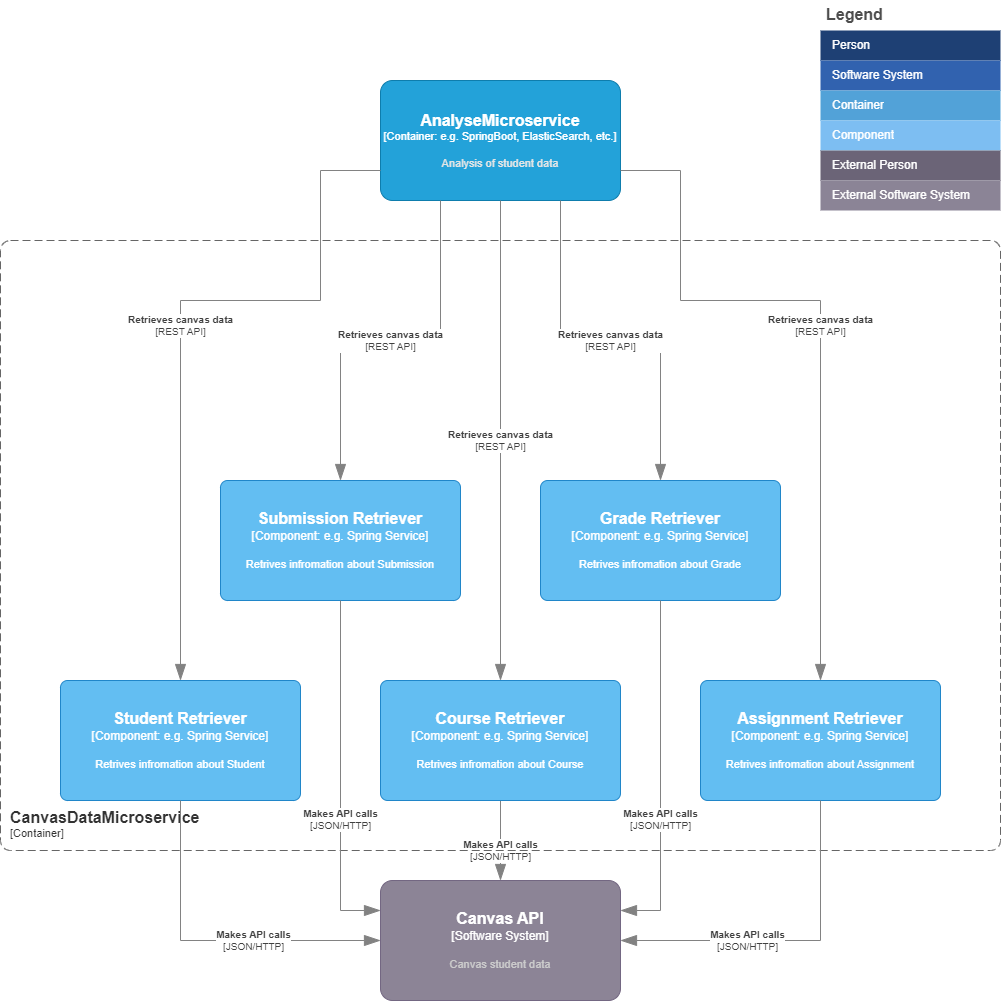
# Level 1: System Context diagram

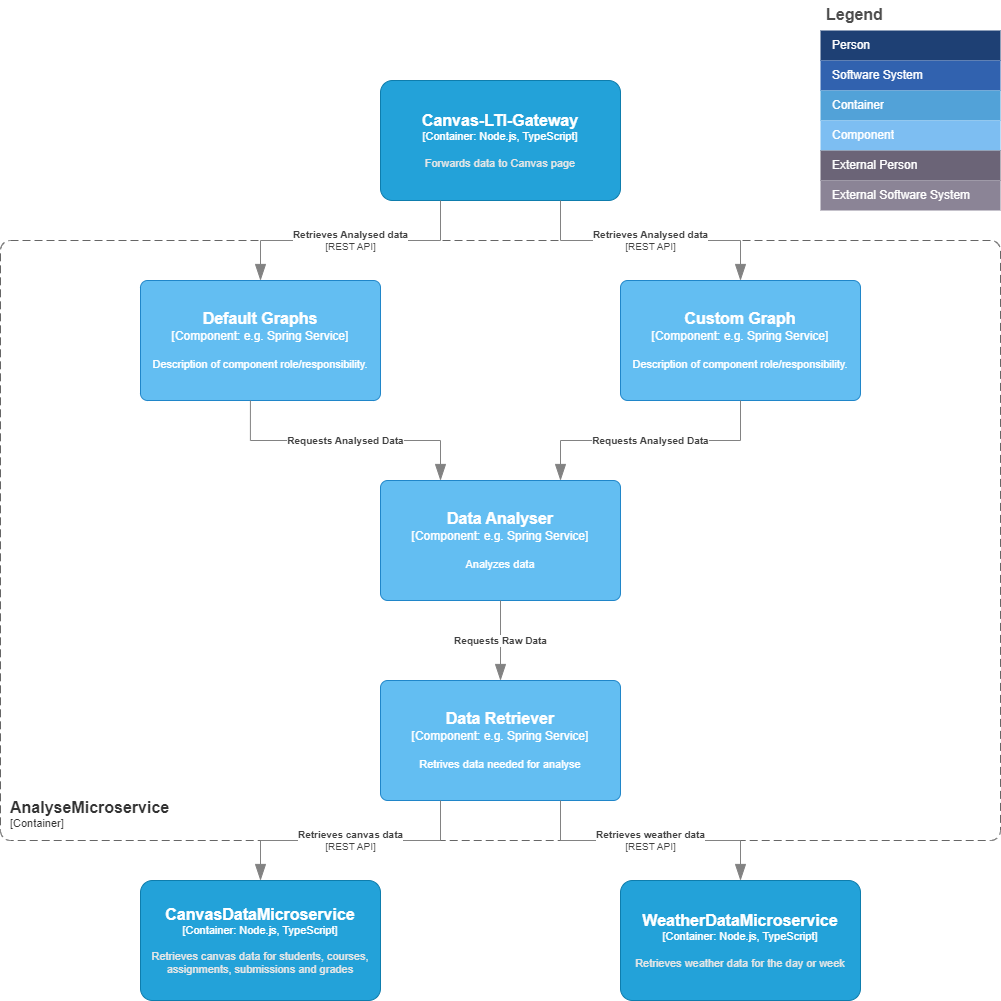


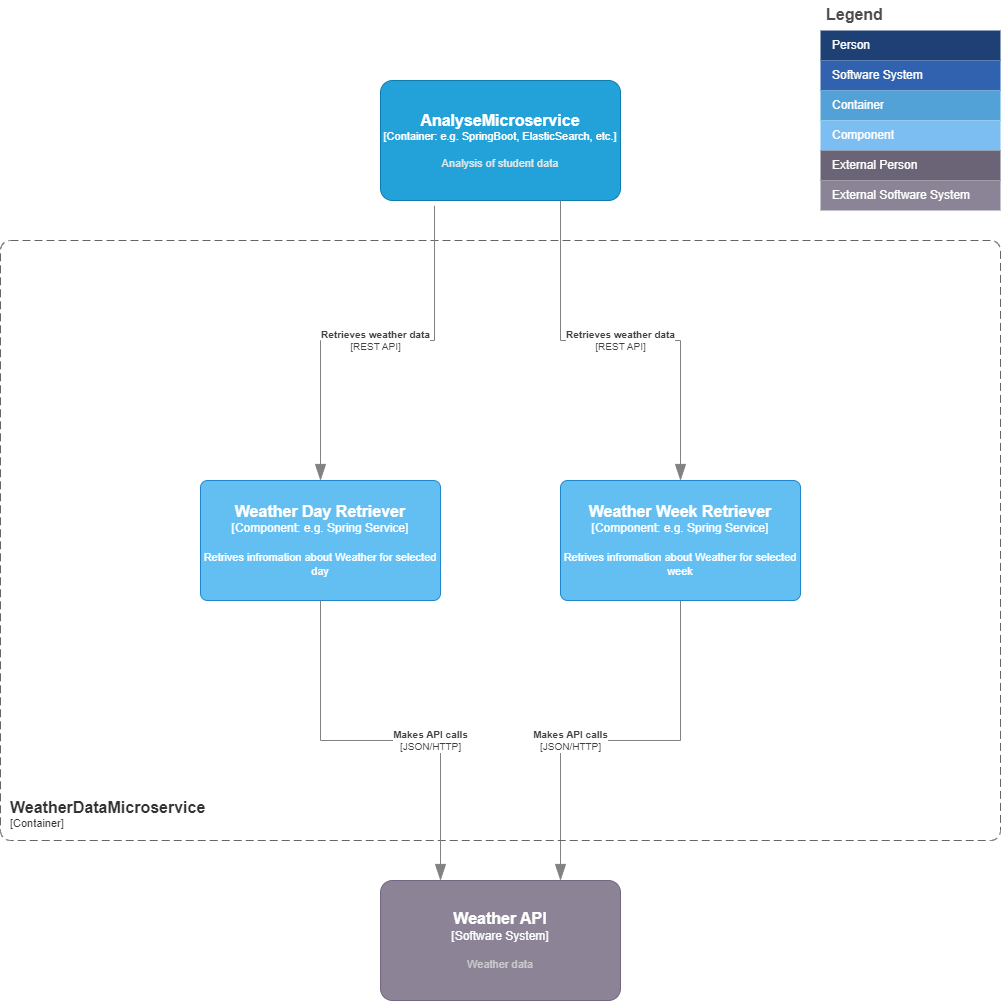
# Level 2: Container diagram



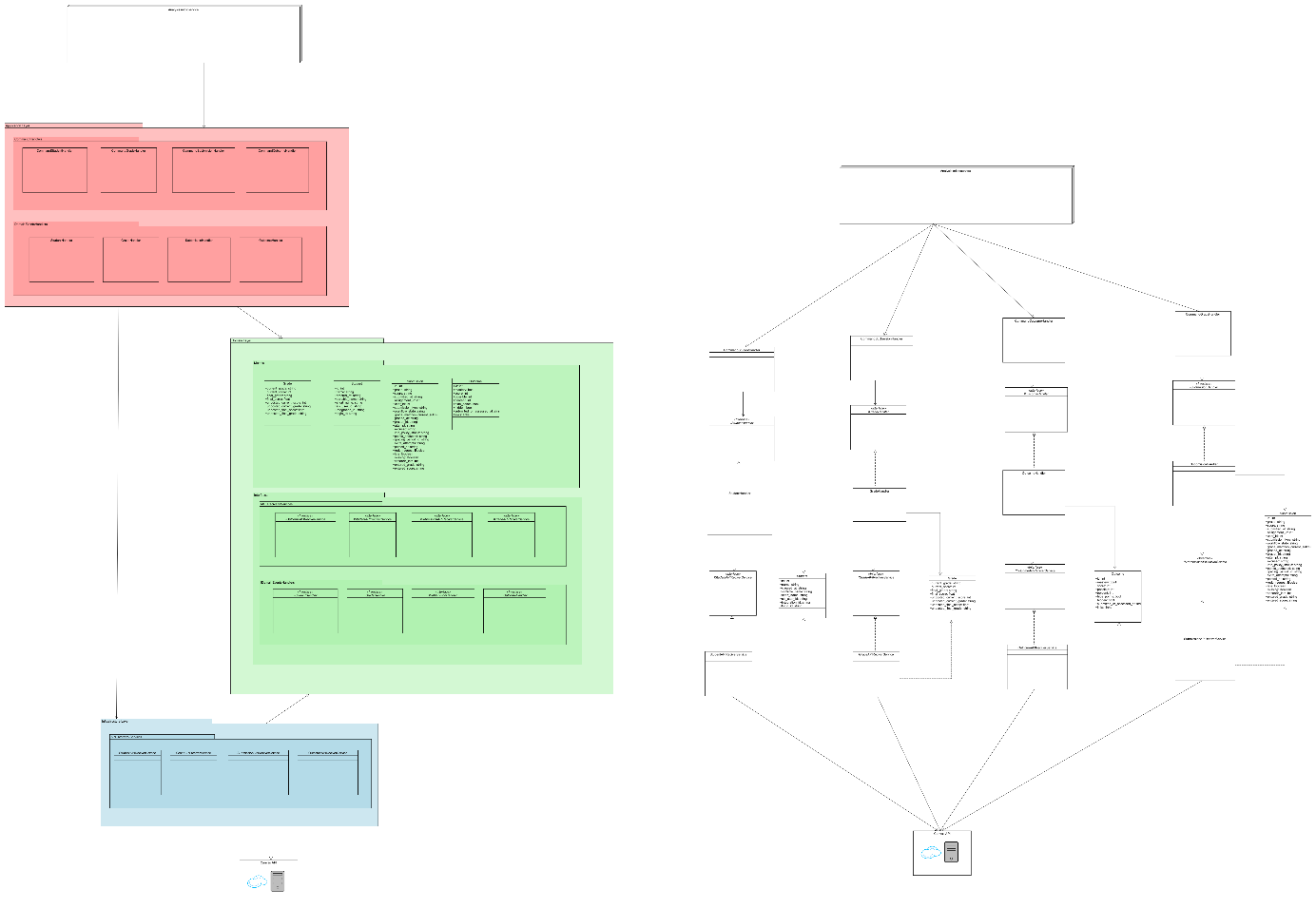
# Level 3: Component diagram

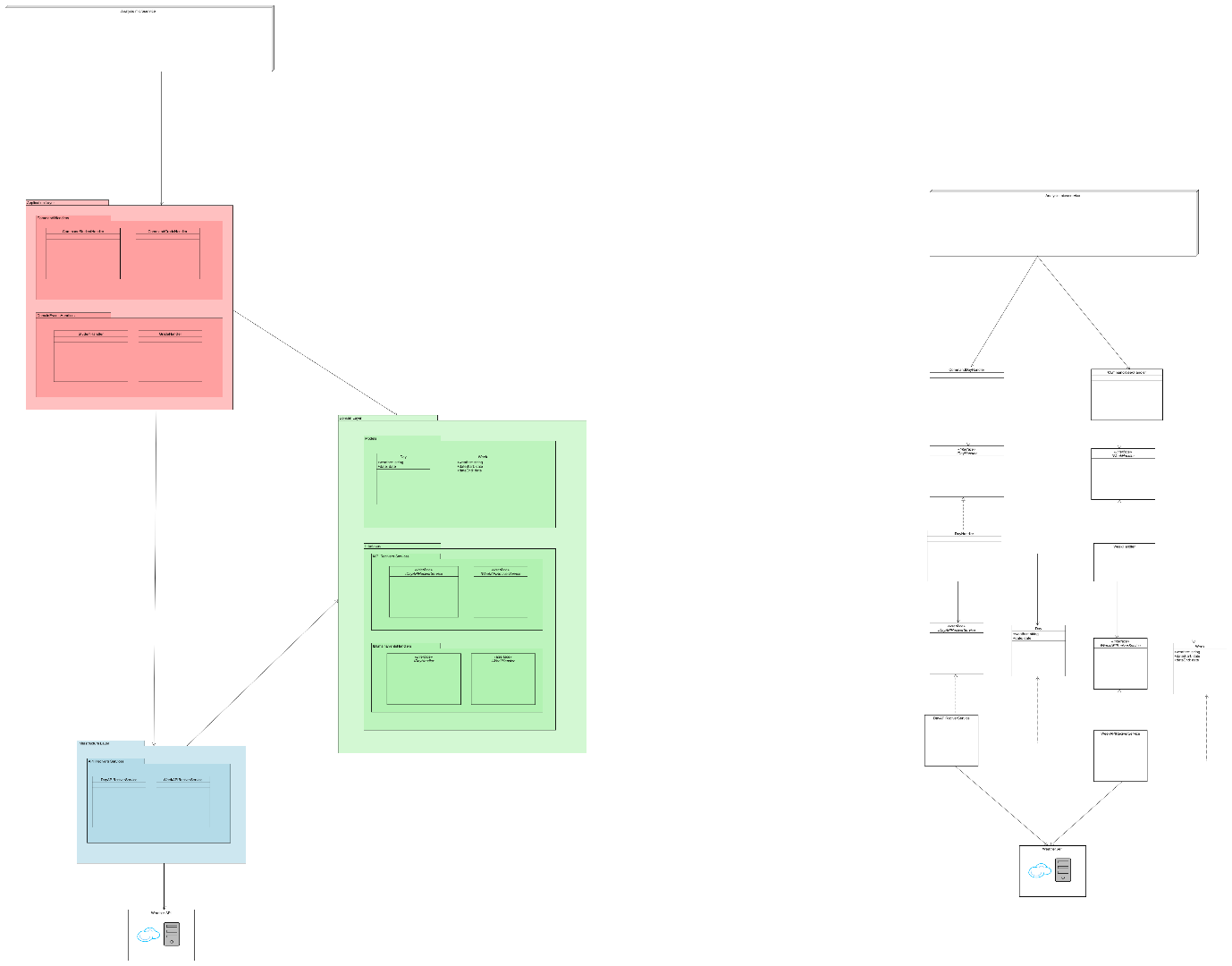






# Level 4: UML





# Architecture of Microservices

## Canvas Dashboard Microservice

DDD

## Canvas Dashboard Gateway Microservice

Gateway Microservice is responsible for capturing API calls from Canvas Dashboard Microservice, decomposing them and forwarding to appropriate microservices. Benefits of this approach is:

* Better Client Performance (UX) – filters all of the individual requests to the microservices through the gateway which reduces latency.
* Security – less susceptible to DoS attack. It is another layer of security, API endpoints of the other microservices are not wide open to anyone on the internet, with that it is easier to add authentication and authorization for another layer of security.
* Protocol Translation (SSL termination) – gateway allows for usage of HTTPS for communication between Gateway and Client, from Gateway to Microservices just HTTP, which does not require encryption, therefore it is more efficient
* Common functionality offload – part of the business logic is extracted from microservices and is put in the Gateway, which makes other microservices focus on their tasks.
* Clear code – gateway enables extracting code responsible for API routing from frontend to one place. It makes it easier to manage different possible API data retrieve routes
* Measuring Data Flow – gateway enables easy way of measuring data flow and traffic in our application
* BFF (Backends For Frontends) – in future to scale project, API gateways can be implemented for different clients (web app, mobile app, desktop app, 3rd party app).

[*https://microservices.io/patterns/apigateway.html*](https://microservices.io/patterns/apigateway.html)

[*https://medium.com/design-microservices-architecture-with-patterns/api-gateway-pattern-8ed0ddfce9df*](https://medium.com/design-microservices-architecture-with-patterns/api-gateway-pattern-8ed0ddfce9df)

[*https://learn.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern*](https://learn.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern)

[*https://www.youtube.com/watch?v=hWRRdICvMNs*](https://www.youtube.com/watch?v=hWRRdICvMNs)

## Analyse Microservice

DDD

## Weather Data Microservice

DDD

## Canvas Data Microservice

DDD

# Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| TBD | To be discussed |
| QS | Quantified Student |
| LTI | The integration technique that we will use to add an application to Canvas |
| DataAnalyseMicroservice | Main part of QS that is gathering and processing data |
| Microservice | Small independent services that communicates with Application |