**Documentation Sprint A**

**ARQSI**

Patryk Pas

Juuso Jalava

Miika Kauppinen

Benjamin Moreira

Liam Saliën

Contents

[**Diagrams** 3](#_Toc118658973)

[Level 1: 3](#_Toc118658974)

[*C4 - Context Diagram* 3](#_Toc118658975)

[*4+1: Logic view* 4](#_Toc118658976)

[*4+1: Process views* 4](#_Toc118658977)

[Level 2: 8](#_Toc118658978)

[*C4 - Container Diagrams* 8](#_Toc118658979)

[User 8](#_Toc118658980)

[Role 8](#_Toc118658981)

[Warehouse 9](#_Toc118658982)

[Delivery 9](#_Toc118658983)

[Package 10](#_Toc118658984)

[Truck 10](#_Toc118658985)

[Route 11](#_Toc118658986)

[*4+1: Logic View* 11](#_Toc118658987)

[*4+1: Implementation View* 12](#_Toc118658988)

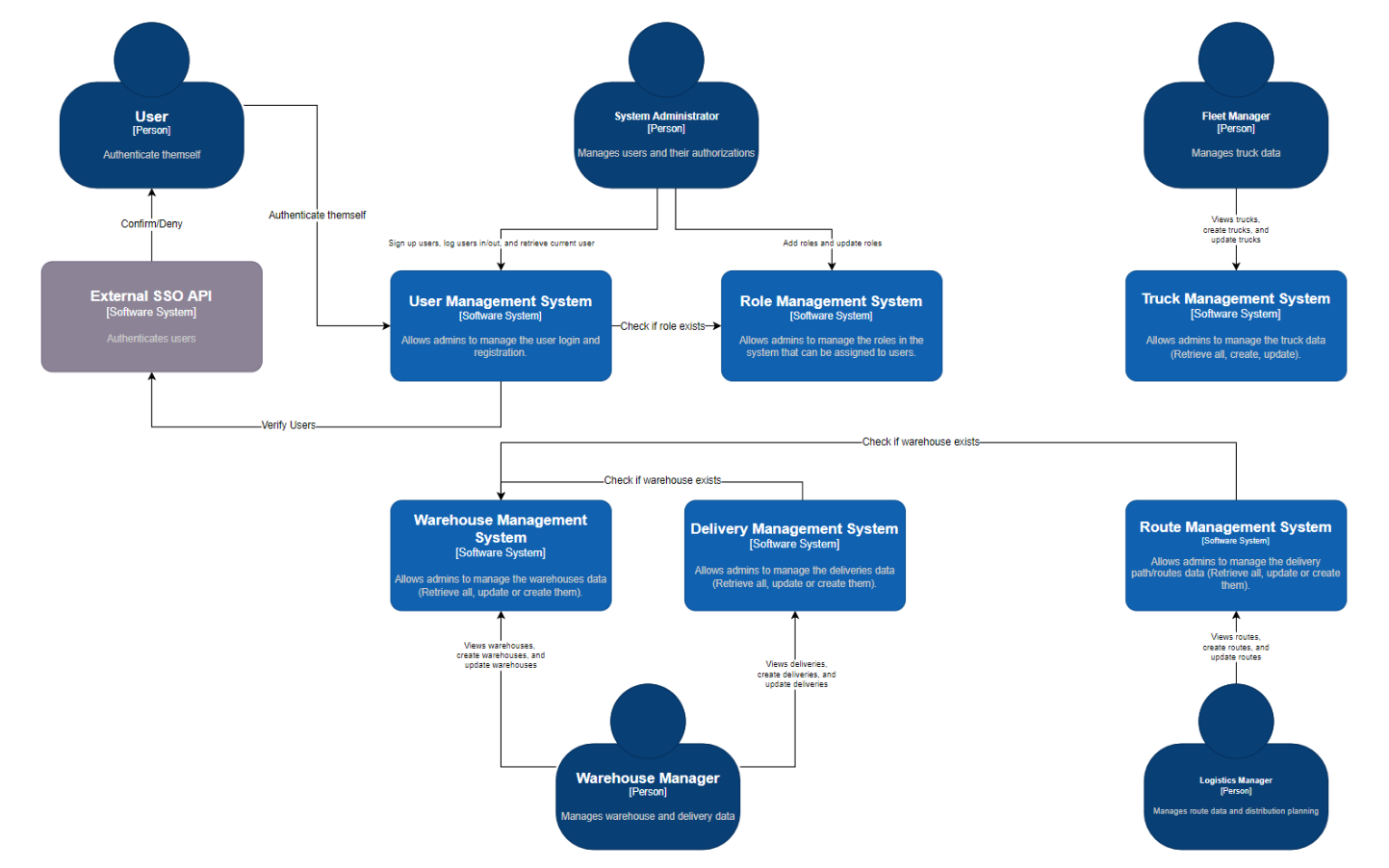
[*4+1: Physical View* 12](#_Toc118658989)

[Quick Design 13](#_Toc118658990)

# **Diagrams**

# Level 1:

## *C4 - Context Diagram*



We see that various (management) Systems are integrated within each other and therefore are also dependent on each other. For example, the Delivery and Route Systems can’t function without the Warehouse System, because they are dependent on it to check if the warehouseIds that are stored in their own Systems are still valid/still exist.

## *4+1: Logic view*

Diagram

Description automatically generated

There are 5 actors that will interact with System:

System admin, Fleet/Warehouse/Logistic Manager, and a regular User.

The ElectricGo System consists of multiple containers and components, and interacts itself with Single Sign On API that makes it possible for users to authenticate themselves.

## *4+1: Process views*

1.

A picture containing diagram

Description automatically generated

This Sequence Diagram shows how data should be retrieved within the ElectricGo System. A user will do a Request to the System to retrieve the requested data, and the System will respond with data or errors.

This is the same for retrieving trucks, deliveries, routes, and packages data.

2.

Diagram

Description automatically generated

This Sequence Diagram shows how to create/POST route data when the System is dependent on another System (in this case Warehouses). It works the same for deliveries. We assume in this scenario that warehouses will always return successfully.

3.

A picture containing application

Description automatically generated

This Sequence Diagram shows the process that a User will go through when registering an account into the System and then trying to log into it with the help of the Single Sign On System.

For the registration to be successful, the data should match the required attributes (= name, email, password) and the role(id) should be valid.

For the log in to be successful, the credentials (email and password) should be valid and known to the system.

4.

Table

Description automatically generated

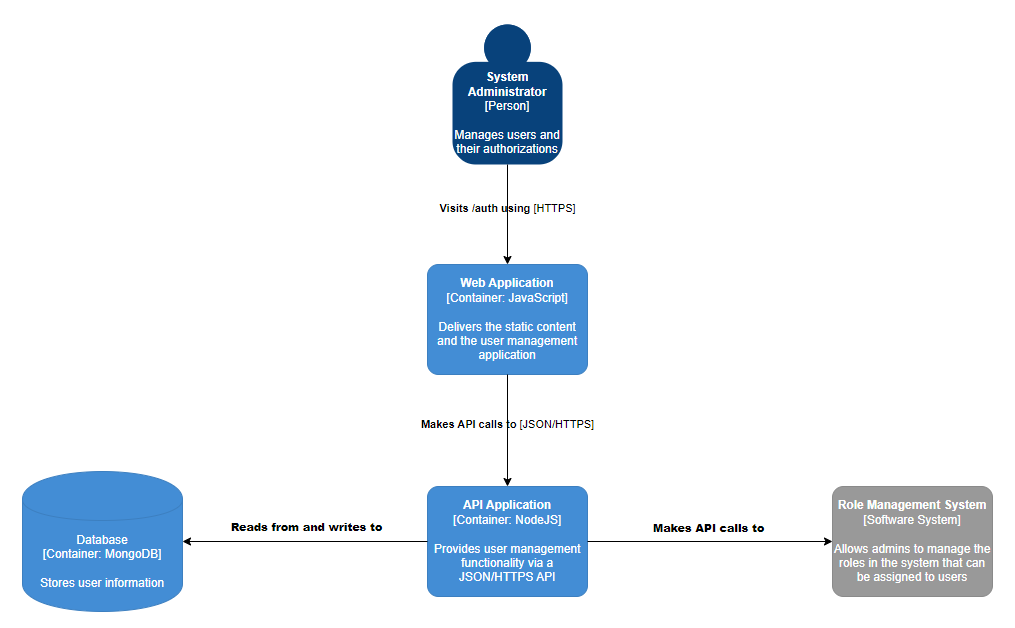
This Sequence Diagram shows how data should be updated within the ElectricGo System. A user will do a Request to the System with a specific id and (in this example) truck data, after which the System will first check if it know the truck, and afterwards if the id given in the truck data matches this of the truckId to which the request was sent.

This is the same for retrieving warehouses, deliveries, routes, and packages data.

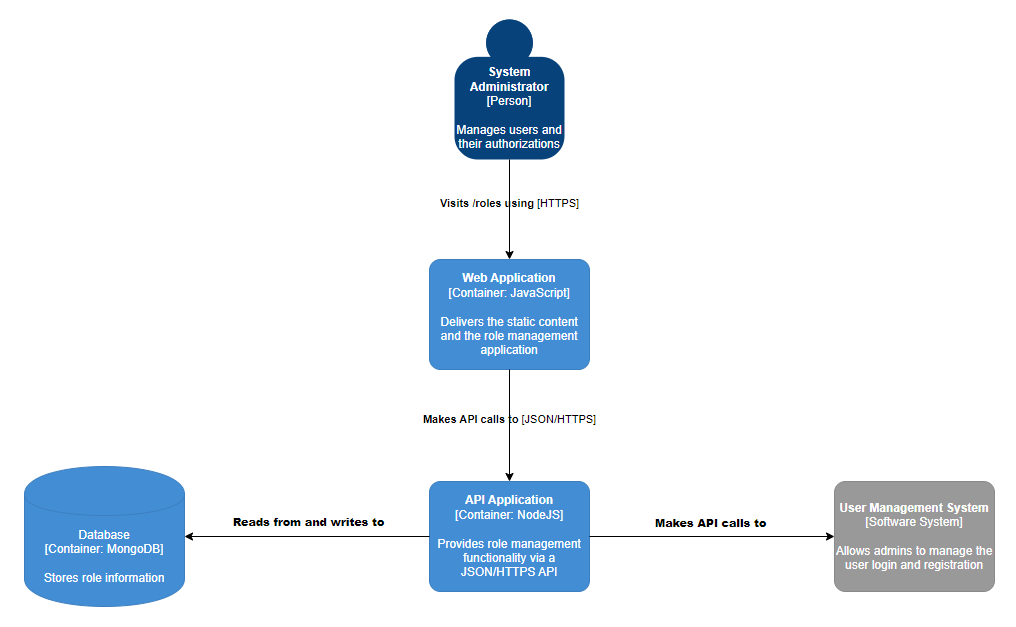
# Level 2:

## *C4 - Container Diagrams*

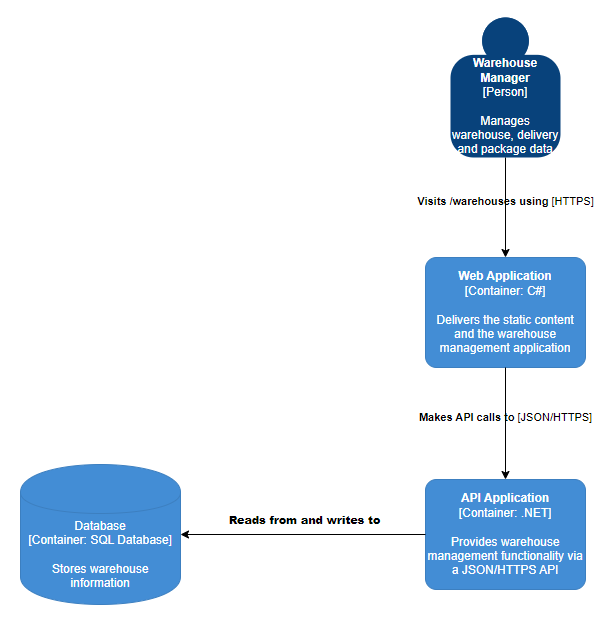
### User



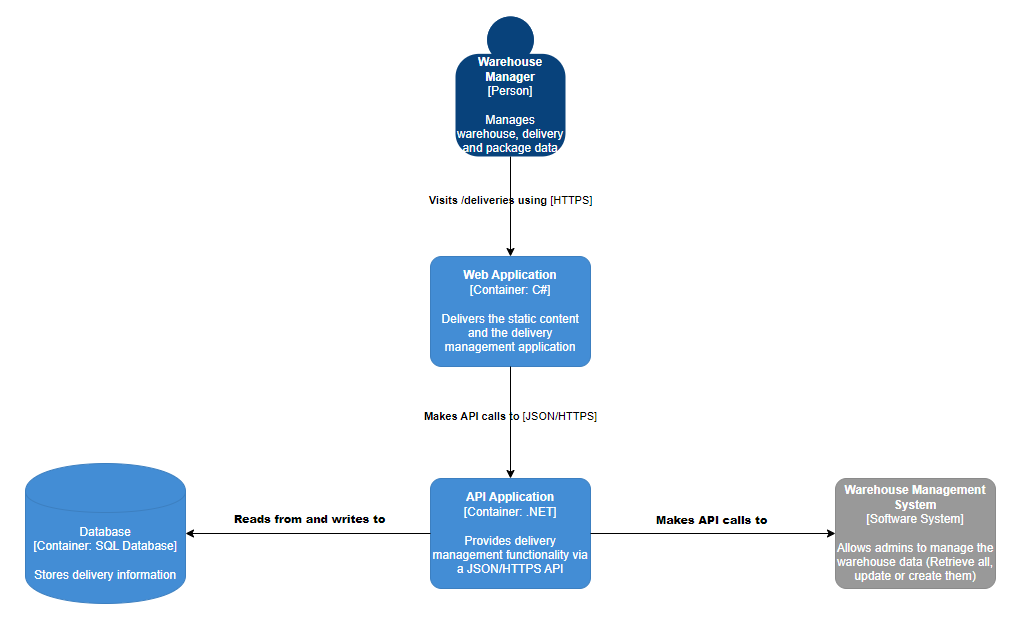
### Role



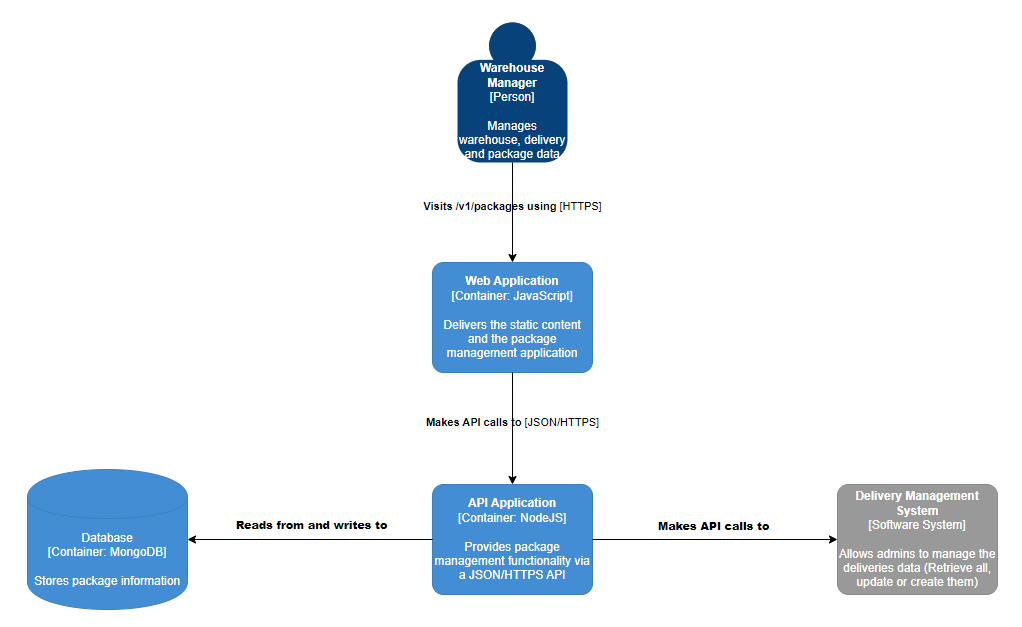
### Warehouse



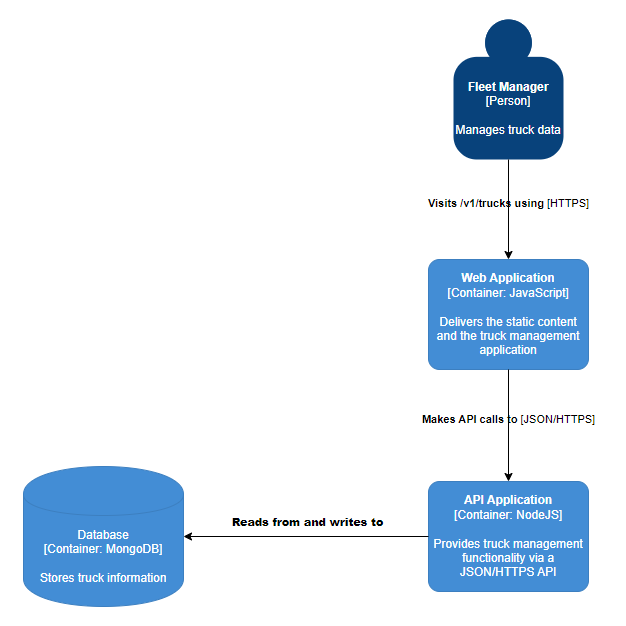
### Delivery



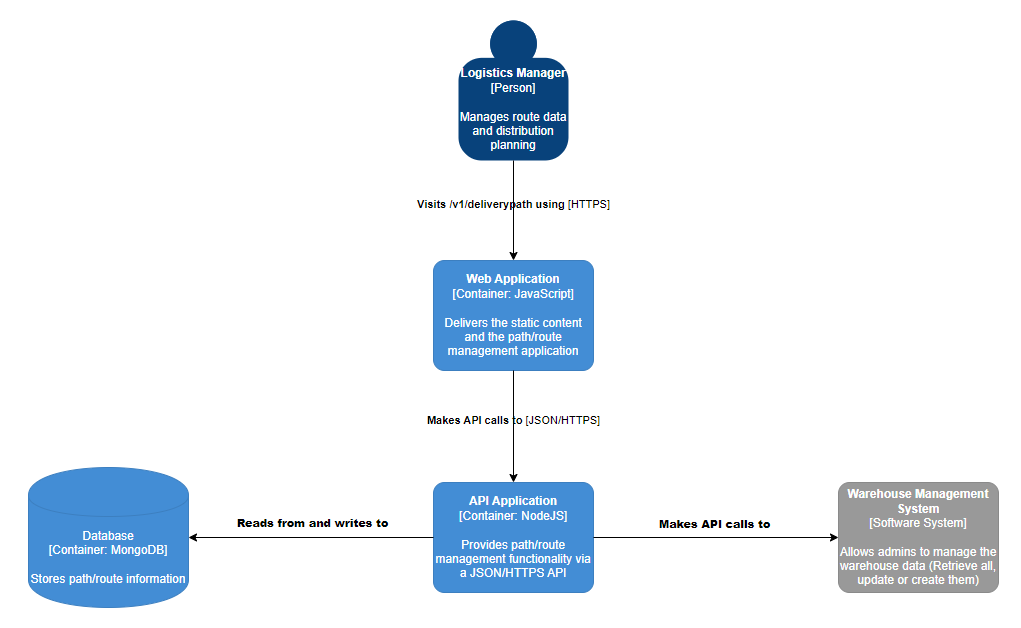
### Package



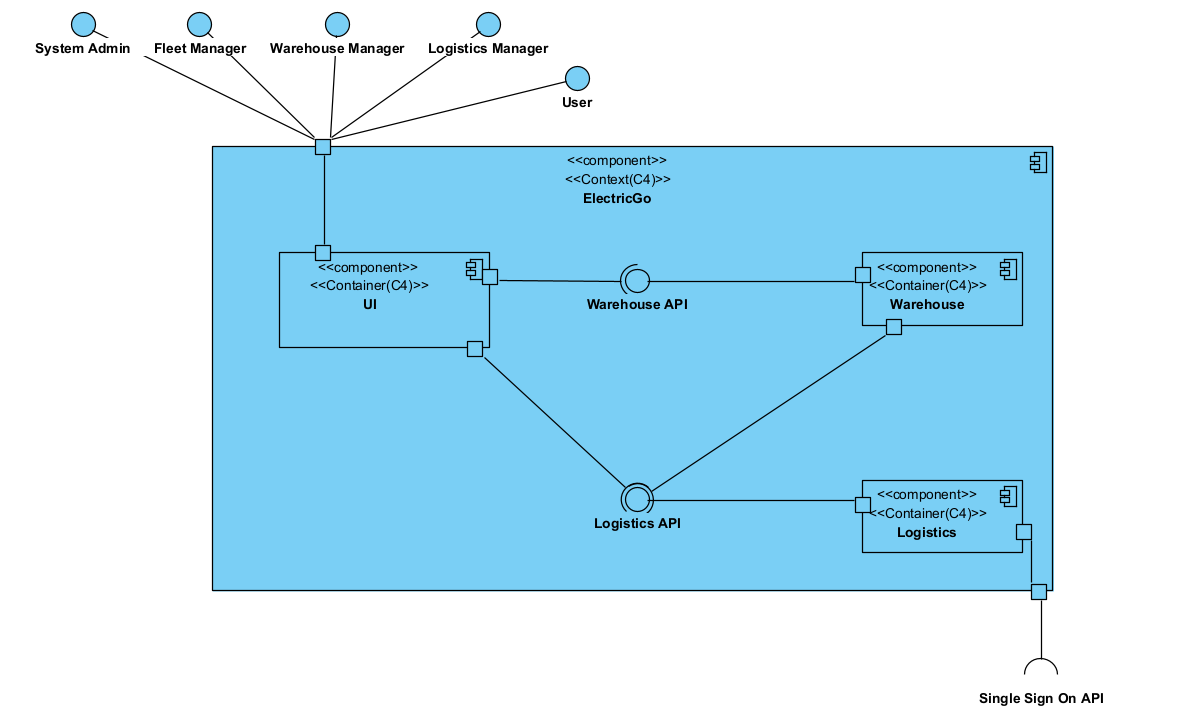
### Truck



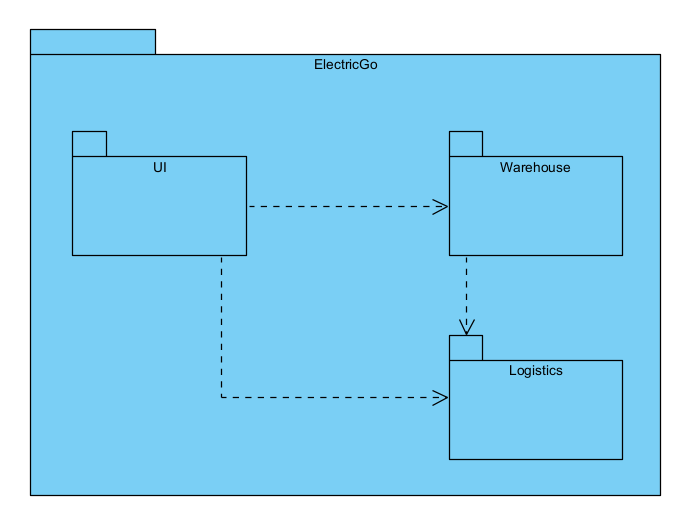
### Route



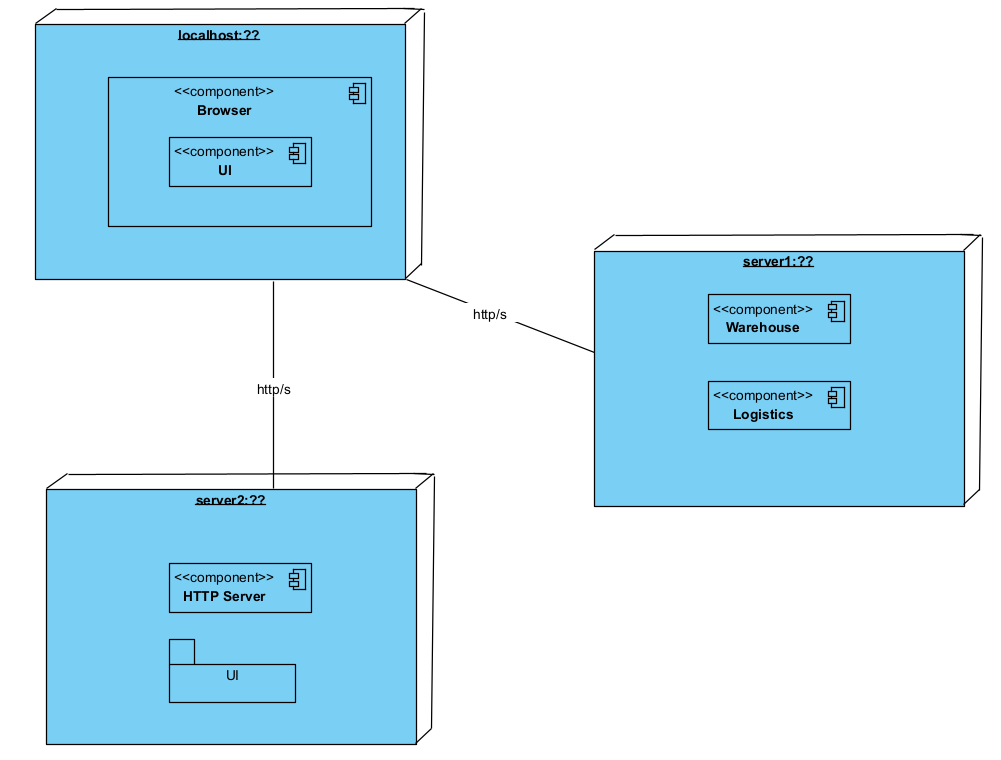
## *4+1: Logic View*



## *4+1: Implementation View*



## *4+1: Physical View*



# Quick Design

When something changes the whole system should be reviewed again, to determine if new dependencies occurred or changes have to be made.

These changes can apply to the processes (e.g. Sequence Diagrams) which might alter the way requests are being handled or executed, and therefore we should keep those possible changes in mind. A way to solve this in the nearby future is by adding opt (alternative for the alt) causes, so that we don’t have to change the diagrams drastically.

We also have to review the other components/containers in detail to see that changes didn’t break any parts of it. For example, if a database is removed, this might have a lot of consequences for all the System parts that were using it. A way to resolve this is by implanting a new database or a valid alternative, or be re-designing the whole System.