The Sebastian (GDS)

1. The system is an online solution for the flight carriers. The idea is to have one system conducted of the flights offers from all the interested companies. This is of course a big convenience for the customers because it reduces the need of visiting each web page of the single flight company. The system has two parts. Web application for customers and desktop application for travel agencies. By using the system, a customer will be able to see departure and arrival schedule for a concrete airports, book the flight, see the booking and cancel it.

2. SAD

a)

1. Non-functional requirements are the characteristics of the system often referred as "iliteis". The main attributes are security, reliability, maintainability, scalability and usability. Non-functional requirements define how a system is supposed to be. For example: a system should be fast and secure. They do not provide the details of how the system should be implemented. Specifying the most important non-functional requirements helps with choosing proper architecture style for the software under development.
2. 1. Performence
   Since the software is intended to be an online booking system it has to perform well. There will be a lot of flight carriers offering their flights. It means that database will have a lot of records.

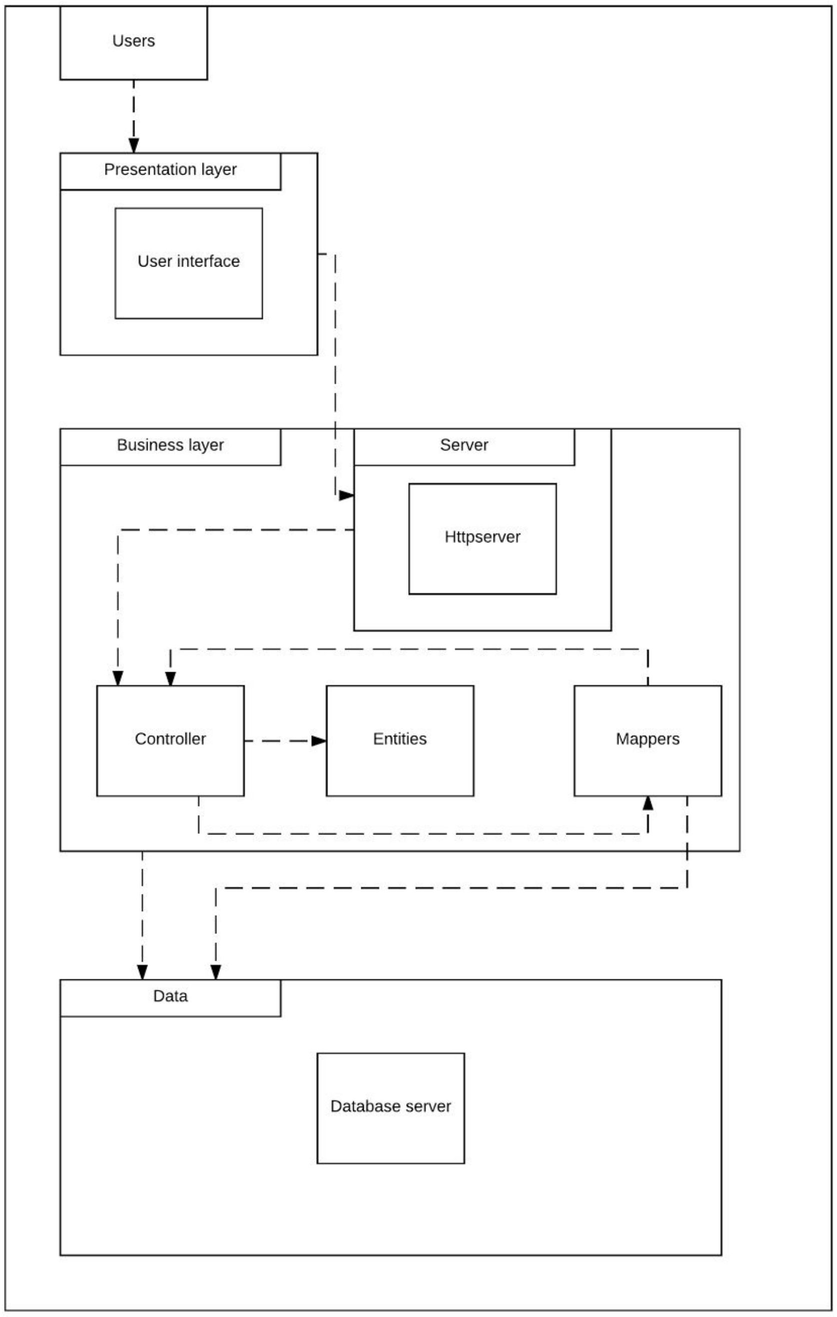
   2. Securtiy
   A lot of important data will be traveling through the system. This calls for ensuring security constraints.

   3. Safety
   In the services like online booking, it is highly possible that a lot of people will complain about canceling reservations. This means that the companies always need to have an access to the relevant reservations data. Backups must be made to ensure data consistency.

   4. Design

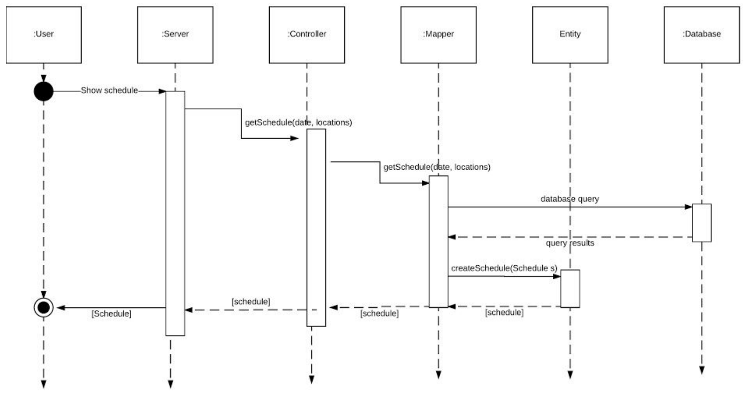
   Customers around the world are using different computers and software. There are numbers of web browsers. The web app has to be able to run on majority of them.

b) 1. Package diagram



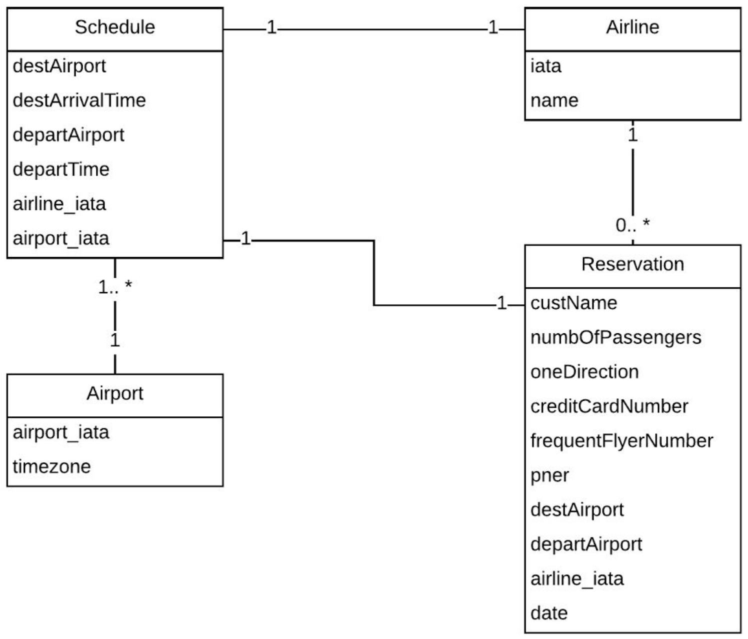
2. The user interface is responsible for presenting data to the user. It also triggers the flow of the data in the system basing on the action performed by the user.
Business layer is the middle tier in the application. The request is sent from the user/agency application. Server deals with the request and calls the methods from controller. The controller is responsible for instantiating entities and calling mappers method. Mappers are responsible for getting and persisting data to the database.

3. Dynamic model (sequence diagram)



4. Code of the implementation is on github.
The implementation itself is not finished. The scenario represents the offline solution for the travel agency. It is possible to make a reservation. After persisting reservation to the database, the pnr is returned so that user can see the reservation and cancel it.
For the online solution, the only missing part is the server that would take the requests from the webapp and call the controller in the backend.
But it shows the idea of the architecture model. There is low coupling between the system elements. Each package has clearly stated responsibilities and each class does exactly what it should do.

c)



d)
The presented solution is three-tier architecture with the mvc pattern. Mvc is the abstraction. By using this model, every component in the system is isolated and performs different operations. This approach makes the components more reusable and logical. The three-tier architecture represents different physical elements in the system.