*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Project Title: **Next Generation Airline Data Exchange Simulator 1.0**

**Team Members: Luis Herrnsdorf, Isaac Reynaldo, Gonzalo Mera**

**Product Owner(s)**: Dr. Shu-Ching Chen, Dr. Miguel Alonso, Steven Luis

**Mentor(s)**: The faculty and graduate students at the FIU AIRlab

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*In a fast-growing Travel Industry, the NDC program was launched by IATA for providing the market with a new adoption of an XML-based data transmission standard called the NDC standard. This standard has been instrumental for enhancing the communication between Airlines and Travel Agents due to its open nature and widespread Airline support. The Next Generation Airline Data Exchange Project was launched by Florida International University to investigate and implement an efficient, open source NDC centric architecture. As the project was being developed it evolved from a monolithic architecture to a microservices one, separating each NDC transaction to individual microservices with one purpose, thus reducing coupling and increasing cohesion. The following project documentation will not only work as a teaching tool but as a walk through of the thought process that lead us moving away from a monolithic architecture to a Microservice one...*

**Table of Contents**

**Introduction** ………………………………………………………………………….………………………………………………….... 5

Current System ……….…………………………………….…………………………………………………………………….. 5

Purpose of New System ………………………………………………….……………………..……………………………..5

**User Stories**

Implemented User Stories ……………………………………………...……………….…………………………………. 7

Pending User Stories …………………………………………………………………...………………………………..….. 10

**Project Plan**

Hardware and Software Resources ………………………..…………………………………………………….… 12

Sprints Plan ……………………………………………………………..………………………………………………………. 13

*Sprint 1*  …………………………………………………………………………………………………………………………... 13

*Sprint 2*  …………………………………………………………………………………………………………………………... 13

*Sprint 3*  …………………………………………………………………………………………………………………………... 14

*Sprint 4*  …………………………………………………………………………………………………………………………... 15

*Sprint 5*  …………………………………………………………………………………………………………………………... 16

*Sprint 6*  …………………………………………………………………………………………………………………………... 17

*Sprint 7*  …………………………………………………………………………………………………………………………... 18

**System Design**

Architectural Patterns …………………………………………………………………………………………………... 20

System and Subsystem Decomposition ……………………………………………………………………………... 21

Deployment Diagram …………………………………………………………………………………………………….…... 22

Design Patterns ………………………………………………………………………………………………………….…..... 22

**System Validation**  ……………………………………………………………………………………………………………………...23

**Glossary**  ………………………………………………………………………………………………………………………………….37

**Appendix**  ………………………………………………………………………………………………………………………………….38

Appendix A - UML Diagrams ………………………………………………………………………………………………. 38

*Static UML Diagrams*  ……………………………………………………………………………………………………….38

*Dynamic UML Diagrams*  …………………………………………………………………………………………………..40

Appendix B - User Interface Design …………………………………………………………………………….…... 52

Appendix C - Sprint Review Reports …………………………………………………………………………...…… 69

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents …………………………………………………………………….……………….... 74

**References** ……………………………………………………………………………………………………………………...…………80

# Introduction

As the world approaches a Post-Brick-And-Mortar future the tech industry is influencing the way customers approach the purchase of plane tickets. Online Travel Agents are substituting the traditional Travel Agent experience and allowing Airlines more control over the distribution of their product. Because of these developments the International Air Transport Association or IATA has developed an XML-based data transmission standard called New Distribution Capability standard (NDC) which is revolutionizing the Travel Industry. The NDC Standard enhances the capability of communications between airlines and travel agents and is open to any third party, intermediary, IT provider or non-IATA member, to implement and use.

## 

## Current System

There’s no open source system that does what this project has set to do. As far as open source NDC applications go, open-ndc.org has different tools like NDC syntax validator and other useful tools which have helped us in the development of the project. But there’s no actual open source system that implements a Google Flight like software that empowers the NDC standard. The architecture designed for Next Generation Airline Data Exchange Simulator we believe will evolve through every iteration and will become an extremely useful tool to empower not only Airlines but also Online Travel Agents to adapt the NDC standard in the near future.

## Purpose of New System

The implementation of the new NDC standard enables the travel industry to transform the way air products are retailed to corporations, leisure and business travelers, by addressing the industry’s current distribution limitations: product differentiation and time-to-market, access to full and rich air content and finally, transparent shopping experience. Available on a voluntary and open basis, NDC benefits different industry players across the airline distribution chain. The current standard is 18.2 (2018 second release). Structured around key functional domains, the NDC schemas provide the opportunity to address the end-to-end airline distribution process, e.g. shopping and order management, to deliver enhanced customer experiences. This New System has been designed to provide a blueprint for an effective adoption of the NDC standard for Airlines and Online Travel Agents alike.

# User Stories

The following section provides the detailed user stories that were implemented in this iteration of the Next Generation Airline Data Exchange Simulator project. These user stories served as the basis for the implementation of the project’s features. This section also shows the user stories that are to be considered for future development.

## Implemented User Stories

* **NGDS-20**: As a user I want to see the details of the flight so that I can make a decision to book it.
* **NGDS-26**: As an Airline I would like to receive a shopping request so that I can sell my flights.
* **NGDS-27**: As an Airline I would like to store my flights so that I can access them and sell them.
* **NGDS-28**: As an Airline I would like to find flights that have been requested so that I can send back the flight references.
* **NGDS-29**: As an Online Travel Agent or User I would like to send a query to an Aggregate Server so I can find multiple flights with my input.
* **NGDS-30**: As an Online Travel Agent I would like to Add flights to my Shopping Cart so that I can book them later.
* **NGDS-31**: As an Online Travel Agent I would like to Delete Flights from my Shopping Cart so that I don't book them.
* **NGDS-32**: As an User I would like to create my personal account so that I can keep flight history.
* **NGDS-33**: As an User I want to be able to sign in with my personal account so that I can see my flight history.
* **NGDS-34**: As an Aggregate Server I would like to send a flight request to multiple Airlines so that I find more matches for the query.
* **NGDS-35**: As an Aggregate Server I would like to keep stored recent searches for a determined time so that I can give query response efficiently.
* **NGDS-38**: As a systems architect I want to define the transactional behavior of the different databases so I can identify an optimization strategy that I can use to design and implement my db.
* **NGDS-39**: As systems architect I want to identify the main components of the different systems so that I can draw a high level diagram of the app.
* **NGDS-40**: As a system architect I want to design endpoints for each database so that I can make query and access data efficiently.
* **NGDS-41**: As a system architect I want to research best practices for architectural implementation of the OTAs so that I can find the most efficient method.
* **NGDS-42**: As a system architect I want to research best practices for architectural implementation of the Airlines so that I can find the most efficient method.
* **NGDS-43**: As a system architect I want to research best caching efficiency methods for aggregator services.
* **NGDS-44**: As a system architect I want to research best caching efficiency methods for OTAs's cache, so that I can find the most efficient way to reduce querying requests.
* **NGDS-45**: As a system architect I want to identify behavior of reads and writes for OTA's database usage.
* **NGDS-46**: As a system architect I want to identify behavior of reads and writes for Aggregate Service’s database usage.
* **NGDS-47**: As a system architect I want to identify a strategy to address the reads and writes behavior of the OTA's database usage.
* **NGDS-48**: As a system architect I want to identify a strategy to address the reads and writes behavior of the Aggregate database usage.
* **NGDS-49**: As an Airline I want to sell flights directly to travelers
* **NGDS-51**: As a System Designer I want to have three instances of Airlines so I can test the NDC transactions.
* **NGDS-52**: As a system architect I want my airline database to perform well with around 600 transactions per second of reads
* **NGDS-53**: As a system architect I want to use similar airline data for fares over the same time period and market.
* **NGDS-54**: As a system architect I want to connect the OTA to the Airline endpoint so that I can start connecting the services.
* **NGDS-56**: As a user I want to search a flights from an Airline directly so that I can Book from them.
* **NGDS-57**: As an User I want to store my current flight search so that I can quickly find it again.
* **NGDS-58**: As a Aggregate Server I want to do cashing to flight search so that I don't have to unnecessarily look for a flight straight from the Airline.
* **NGDS-59**: As a system Architect I want to connect to three different Airlines so that I can aggregate the correct flights being requested.
* **NGDS-61**: As a system Architect I want to create a Poster with the process of the project so that I can finalize the Project.
* **NGDS-63**: As a System Architect I want to create a Open Source License so that the project can be exposed to the open source community.
* **NGDS-64**: As a system Architect I want to create documentation of the process of the project so that I can finalize the Project.

## Pending User Stories

* **NGDS-55**: As a system architect I want to implement a Api Gateway so that I can connect the different microservices to it.
* **NGDS-60**: As a System Architect I want to verify NDC schemas so that I know when I'm using the correct protocol, and receiving it.

# Project Plan

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

Hardware:

* Computers in CIS Lab

Software:

* MS Code IDE for writing code.

● Chrome browser for testing local development.

● Postman to test for HTTP requests.

● Google Docs for writing documentation.

● Google Slides for presentation.

● Draw.io for modeling diagrams.

● Bitbucket for version control.

● Jira for backlog and sprint management.

## 

## 

## Sprints Plan (Notes)

### Sprint 1

NGDS:

FrontEnd: OTA/User, we can find inspiration with Travelocity, expedia, hipmunk… etc

As a public user I WANT TO book a flight via web application SO THAT I can travel … etc

…

As an public user I would like to select from the list of flight so that I can purchase this flight.

As a user I want to provide origin departure and date SO that I can get a list of available flights details(carrier, etc…)…

As a user I want to

As a OTA: I want to consume carrier flight data so that I can use/present that to satisfy my public users,

As a carrier: I would like to publish my flight information so that other OTAS can consume this information to provide bookings.

BackEnd: building an NDC api.

Origin, destination, date ,time, one way or round trip. On the front end, to match with the database.

### Sprint 2

Sprint 2 planning:

-UserStories:

-ShoppingCart(Booking)

-Architecture two servers and UI, high level

(Aggregate)

As a public user I need to query multiple airlines for flight info, and see results in aggregated form

-aggregation database server, receive query from ui and make connection with multiple airline requests.

(aggregate server caching queries)

As a public user I would like to see cache query information so I can efficiently review query results.

As an OTA I need to be able to request flight information from one or more airlines and receive the results in NDC format.

As a user I want to filter my search….

(database tech for ui query) 8base -> implement it

### Sprint 3

The purpose of the project is to investigate methods to implement ndc in the airline industry and to study the transactional behavior which will occur when the system is implemented.

Hyphothesis:

Demonstration of what we are testing, an aggregation based service embedded with OTA will provide an efficient mechanism to manage the different airlines that must be presented to the user when they make a query.

Analysis will have both the efficiency of moving data between OTA and airlines and the cost it would be.

\*Research data warehousing techniques that OTA’s use or airlines.

\*Research CQRS architecture

* Behavior - optimization strategy and - design implementation..

\*Identify behavior of reads and writes for each database in the architecture and then identify strategy to take to address this r/w behavior, and then design an implementation, for instance the db for airlines will be a limited number of writes and exponential number of reads, where the aggregator will have both each person can generate thousand of queries, which it will have cached data that you don’t have to redo.

* OTA cached data base is individual request set that don’t want to trigger cascade set of queries but reduce the query impact of the architecture.

User stories

Caching efficiency between aggregator services and OTAs’s cache… : two user stories one for ota and one for aggregator.

- As a systems architect I want to define the transactional behavior of the different databases so I can identify an optimization strategy that I can use to design and implement my db.

- Design endpoint for each database so that I can make query and access data efficiently.

### Sprint 4

90 million transaction per day.. architecture for handling 600-4000 transactions per second.ANALISIS

Q- most used transaction for NDC.

CQRS\*\*\*?

TODO: Diagram for this semester of workflow between OTA and Airline, for ticketing a flight. (U.S.)

Part of the analysis of the data, based on the architecture of the product. If we have 1000 booking REQ per minute. ?1000 booking request for every ticket sold.

FOCUS on ARCHITECTURE\*\*\* STRATEGIES FOR BOOKING AND TICKETING

\*\*\*Caching Strategies for data sets.

Best practices/architecture: high availability systems

NDC find if there’s description for time to live for the data, or how long would it be, and it has to be re requested after the TTL has expired. ---\*\*\* its 20 minutes not for saving whole offers

### Sprint 5

Notes:

Airlines should provide interface that allows OTA aggregate server to acquire NDC transactions. (Luis)

At least three airlines for the OTA to access NDC data. (Luis)

As a system architect I want my airline database to perform well with around 600 transactions per second of reads. (Luis)

As a S.A. we want to use similar airline data for fares over the same time period and market. (Luis)

### Sprint 6

TODO:

- Cashing service in OTA side.

- Three Airlines Aggregation.

- TESTING!!!! Validating NDC some type of schema validation!!!!.

- Poster.

- Documentation.

# 

# System Design

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

This section describes the System Architecture used in the design of the Next Generation Airline Data Exchange Simulator 1.0 platform that was based on the needs and requirements of the product owner, as well as what we have foreseen as possible future scenarios. We present an overview of the components (servers, web and database) and their functions in the system.

The Architecture was divided in two very important pieces:

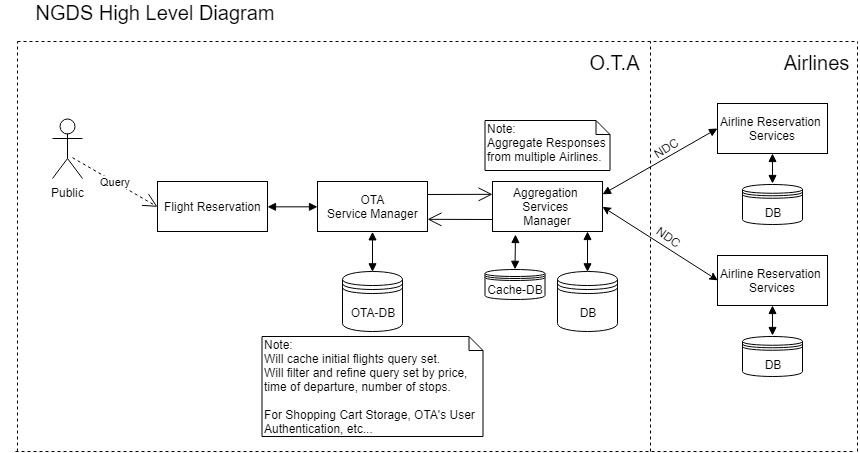
**The OTA (Online Travel Agent)**:

The Flight Reservation Service was focused on User Interactivity, and the Aggregation Service would add all the transactions coming from the airlines. It can also store then in a database and cache them as long as the Airlines permits them. The proposed system architecture for the OTA is a REST pattern which evolved throughout the implementation of the project from a monolithic architecture to a microservice one, to increase scalability and efficiency.

**The Airlines**:

Airlines Reservation Services focuses in the specific Airline which provides with the flights for booking and ticketing. The Airline had its own DataBase and it could be duplicated as many times as possible to do stress Testing on the OTA’s. The proposed System Architecture for the Airline portion of the system is REST Pattern. The larger target use of the platform is at a global level and thus scalability and performance requirements are of first order which is why I choose this architecture. I rely on Nginx native capability to handle large amount of request per second. In an environment where there is a connection of 1 GB/s, 16G RAM and a CPU with 16 cores, Nginx can handle 91,623 requests per second on a 100 KB standard file size. Behind Nginx is ExpressJs framework running on top of NodeJs. ExpressJs will handle the logic between the incoming requests and the database, for which I choose PostgreSQL for its native ability to spawn new workers to handle new connections that ExpressJs send.

## Architectural Patterns

Figure above shows The Architecture of the whole system while in a monolithic state.

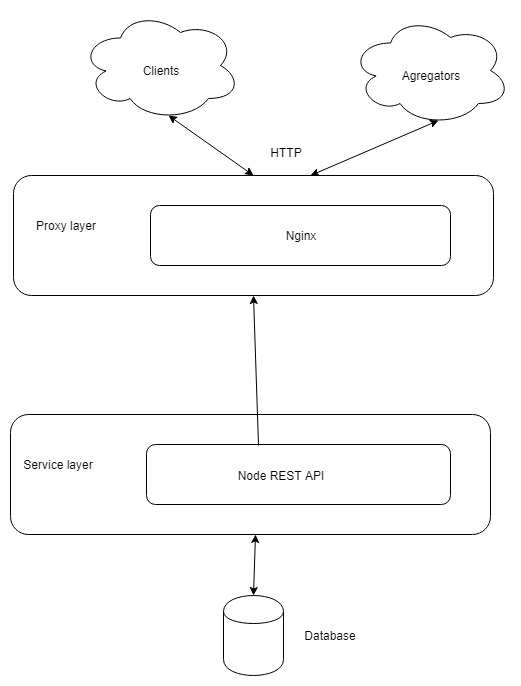


Figure above shows REST Architectural Pattern implemented for the Airline.

## System and Subsystem Decomposition

Clients: Web app that connect to the API to search for flights. #49:Sell flights directly to travelers.

Database: Relational database to store flight related information. #19:Database installation/design.

Service layer: Express/Node backend API to handle HTTP calls. #26:receive a shopping request so that I can sell my flights.

Proxy layer: Reverse proxy that receives HTTP requests and pass them on to ExpressJs. #52:airline database to perform well with around 600 transactions per second of reads.

## Deployment Diagram

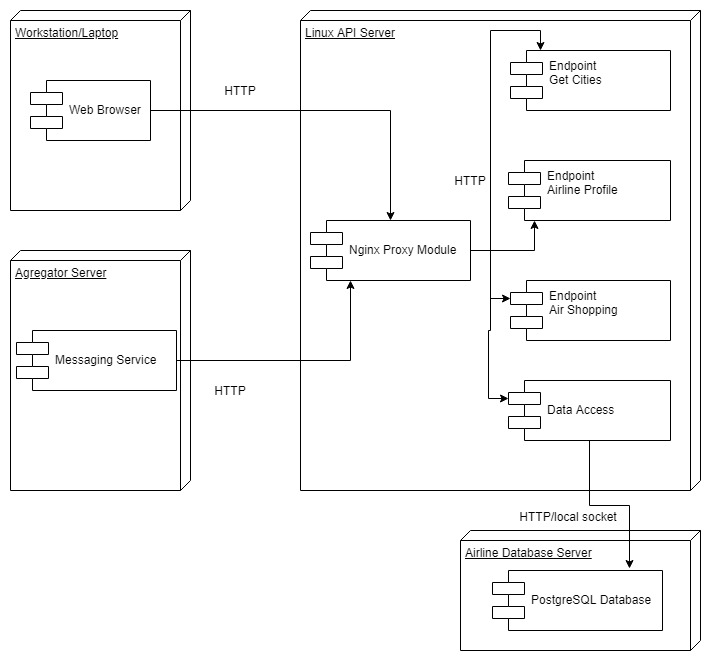


Figure above shows detailed view of Airline Deployment Diagram.

## Design Patterns

# System Validation

Airline:

#26 Receive shopping requests:

* Test case ID: ReceiveShoppingRequests001
* Description/Summary of Test: As an Airline I would like to receive a shopping request so that I can sell my flights.
* Pre-condition: API is waiting for requests and Database have flights data.
* Expected Results: Send list of flights.
* Actual Result: Send list of flights.
* Status (Fail/Pass): Pass.

#49 Sell flights to travellers:

* Test case ID: SellFlightsToTravellers001.
* Description/Summary of Test: As an Airline I would like to sell flights directly to travellers.
* Pre-condition: Traveller request flights from Airline UI.
* Expected Results: Traveller using Airline UI finds a flight to buy.
* Actual Result: Traveller using Airline UI finds a flight to buy.
* Status (Fail/Pass): Pass.

#51 Have three instances of Airlines:

* Test case ID: HaveThreeInstancesOfAirlines001
* Description/Summary of Test: As a System Designer I want to have three instances of Airlines so I can test the NDC transactions.
* Pre-condition: Already finished the development of one instance.
* Expected Results: Send HTTP requests to three different servers IP.
* Actual Result: Send HTTP requests to three different servers IP.
* Status (Fail/Pass): Pass.

#52 Database perform with around 600 transactions per second:

* Test case ID: DatabasePerformWithAround600TransactionsPerSecond001
* Description/Summary of Test: As a system architect I want my airline database to perform well with around 600 transactions per second of reads.
* Pre-condition: Database is up and contains data. ExpressJs can connect to DB.
* Expected Results: System handle 600 requests per second.
* Actual Result: System handle 1000 requests per second.
* Status (Fail/Pass): Pass.

#53 Use similar airline data in databases:

* Test case ID: UseSimilarAirlineDataInDatabases001
* Description/Summary of Test: As a system architect I want to use similar airline data for fares over the same time period and market.
* Pre-condition: First database is already filled with data.
* Expected Results: Receive similar data when querying the databases.
* Actual Result: Receive similar data when querying the databases.
* Status (Fail/Pass): Pass.

# 

# 

# 

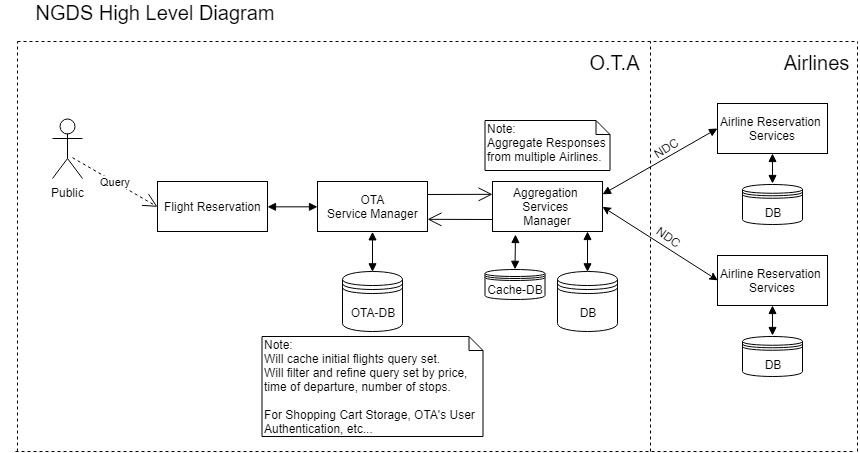
# 

# Glossary

* **Aggregator**: The business function of distributing a Seller’s shopping request to multiple Airlines and aggregating the subsequent responses.
* **Airline**: Supplies product offers in response to receiving a request from a Seller. Airline refers to itself and any subcontracted entity providing a service to the airline.
* **Airline Currency**: Miles, vouchers, residual value EMD, airline issued gift/cash card.
* **Ancillary Services**: or Optional Services Ancillary Services are defined in PSC Resolution 787 as anything outside of product attributes (optional or discounted). Ancillary Services may be bundled in the product offer, or offered as additional, à la carte services. In NDC-related documentation, Ancillary Services is sometimes used interchangeably with the term Optional Services.
* **Anonymous Shopping**: A Shopping Request sent to airlines without Personal Data.
* **API**: Application Programming Interface
* **Authentication**:The process by which a system identifies an individual or a business entity to make sure that the user or the business entity is whom they claim to be, based on attributes that are sent in a message.
* **Cabin**: A compartment where passenger seats are installed.
* **Group Booking**: An Order made under a group name without individual passenger names at the point of creation.
* **Leg**: The operation between a departure station and the next arrival station.
* **Multi Leg Flight**: A flight comprised of more than one leg.
* **NDC**: New Distribution Capability.
* **Offer ID**: The Offer ID facilitates the tracking and verification of individually priced offer(s) selected from the shopping response. Only the Offer IDs of the ORA (Offer Responsible Airline) are returned in shopping responses. The Offer ID is unique to each individually priced offer in the shopping response even if the offer price is zero. The Offer ID may be specific to individual passengers in the offer, and may be associated with a segment or a journey. The set of OfferIDs returned in a response are referenced by a Shopping Response ID.
* **Offer Time Limit**: The time within which offers must be converted into completed Orders.
* **Operating Carrier**: The carrier that holds the Air Operator’s Certificate for the aircraft used for that flight.
* **Order**: An Order is a uniquely identified record of the agreement of one party with another to receive products and services under specified terms and conditions. ‘Order’ supports the sale of a flexible range of airline products and services that are not necessarily journey based (e.g. subscription services). A ‘PNR’, ‘super PNR’ and ‘ticket’ are all today’s versions of airline implementations of aspects of an Order. An Order will contain 1 or more Order Items each with an identifier that is unique within an Airline’s Order Management system. An Order may support non-homogeneity, i.e. each passenger in an Order may hold different sets of order items at different prices.
* **OTA:** Online Travel Agency.
* **PNR**: Passenger Name Record.
* **Price Guarantee Time Limit**: Period for which an Offer price is guaranteed. On expiry an Offer may be re-priced up to the point an accountable document is issued. A Price Guarantee cannot extend beyond the Offer Time Limit unless the Order has been created. Processing of Personal Data Any operation or set of operations which is performed upon Personal Data, such as collection, recording, organization, storage, adaptation, or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, blocking, erasure or destruction.
* **SSR**:Special Service Request.
* **TMC**: Travel Management Company
* **XML**: Extensible Markup Language

# Appendix

## Appendix A - UML Diagrams

Figure above: Monolithic architecture, first iteration.

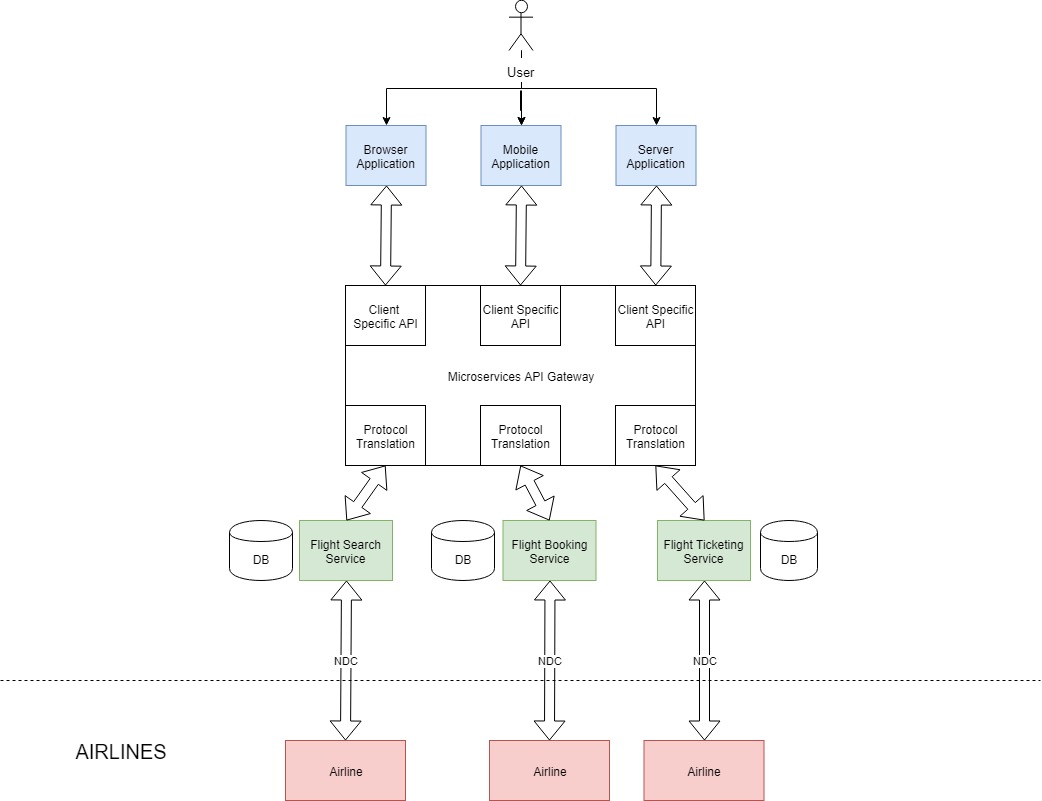
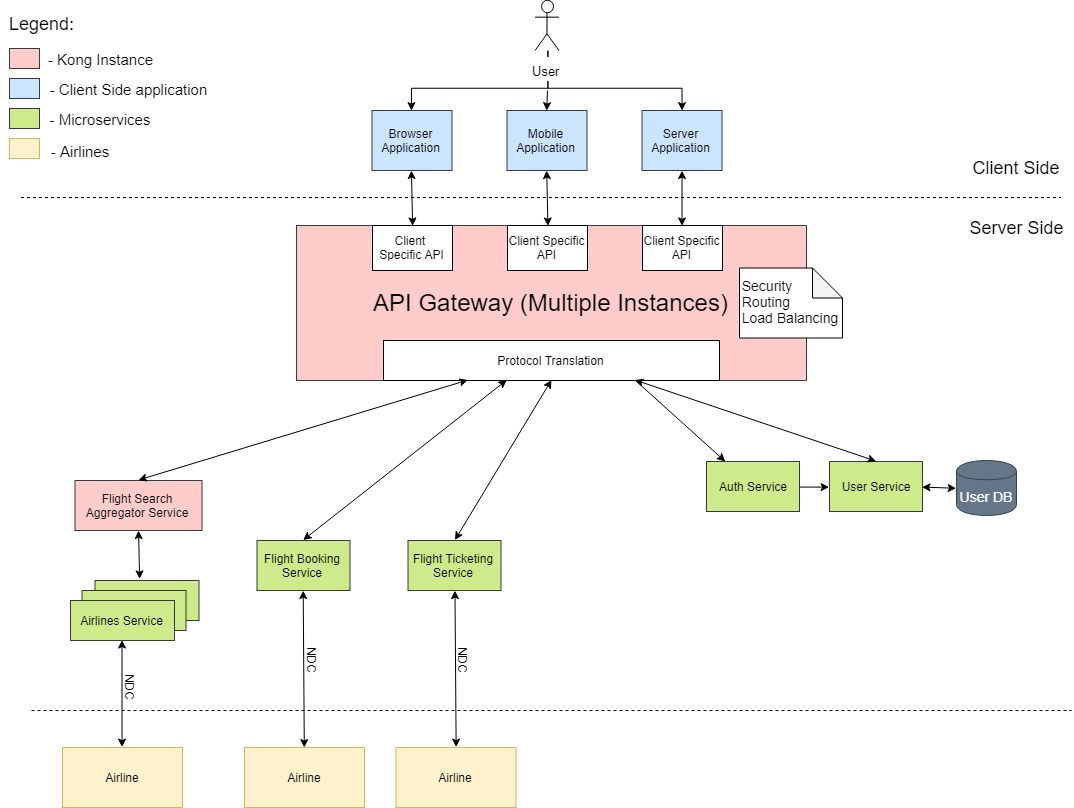
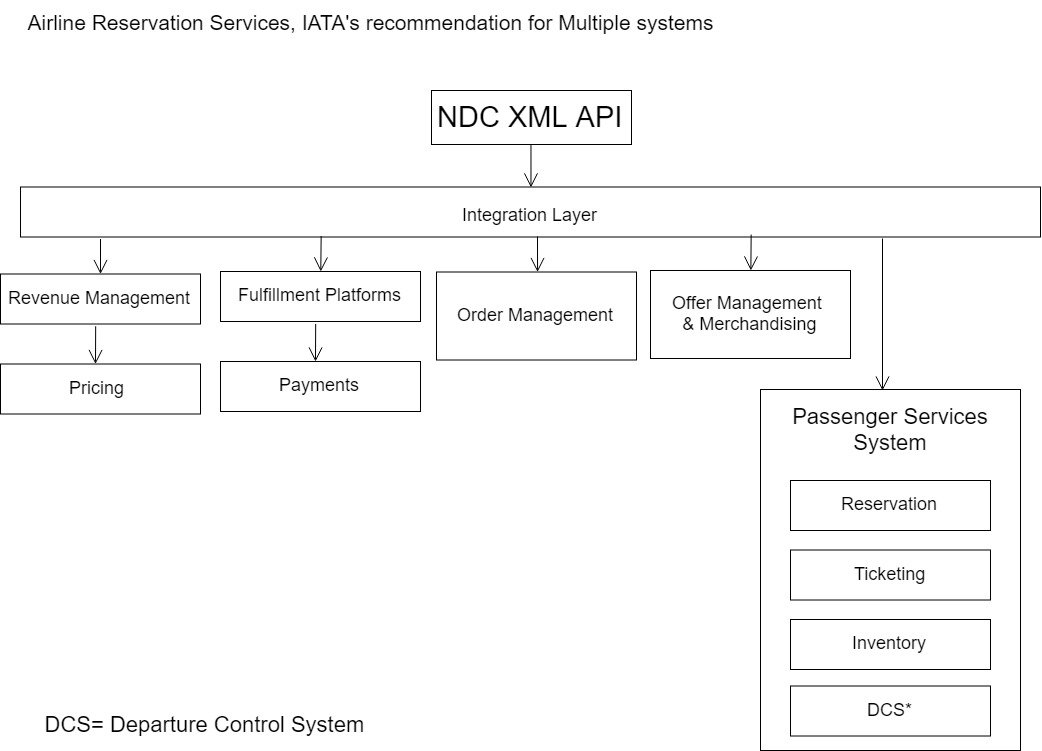
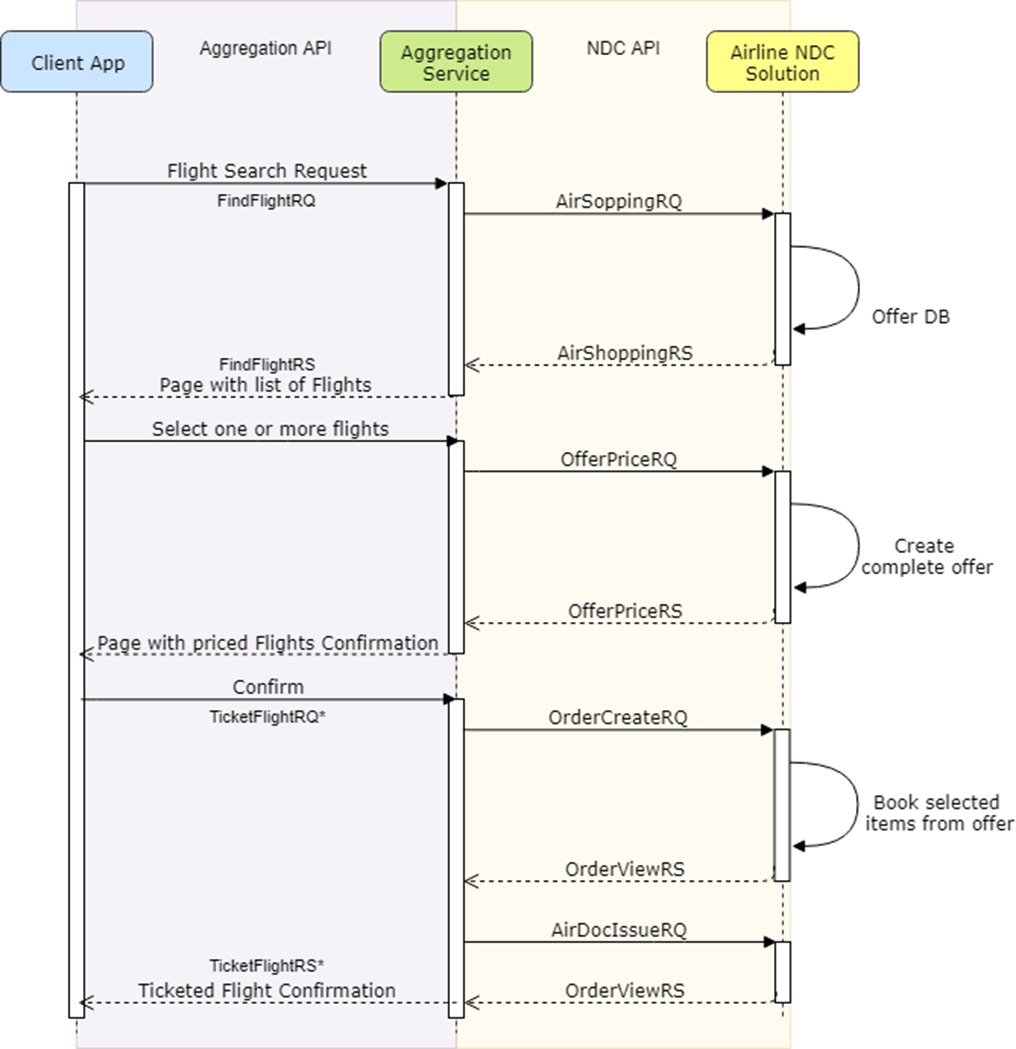


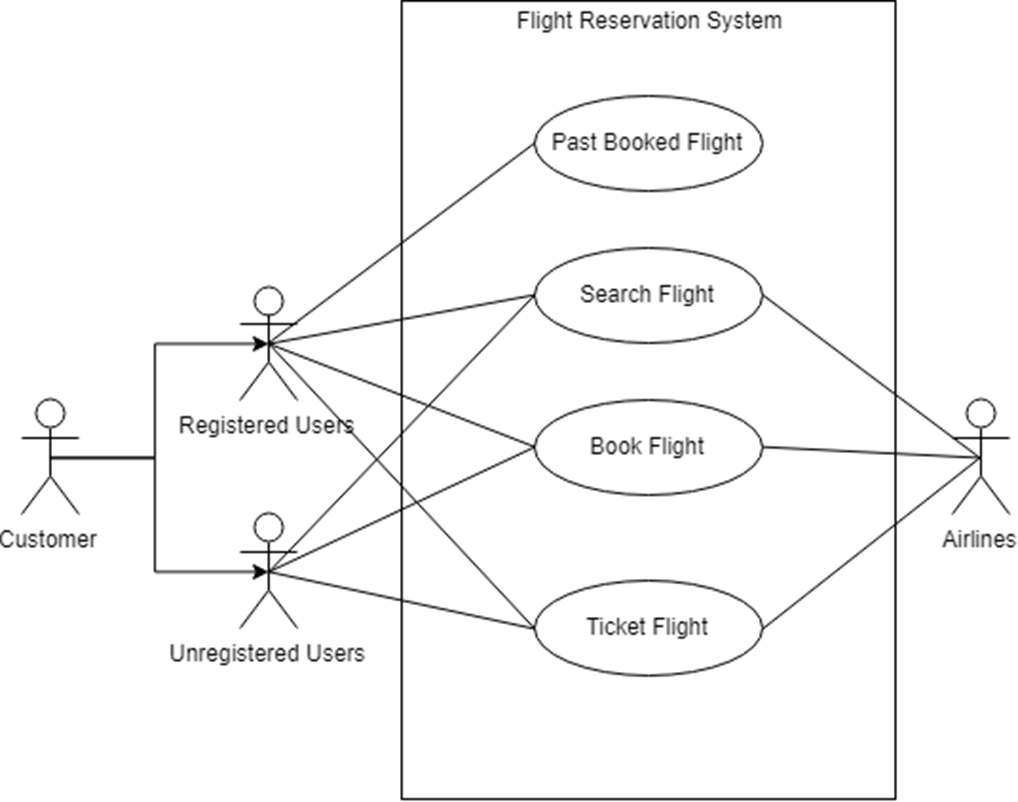
Figure Above: Microservice Architecture v1.0.

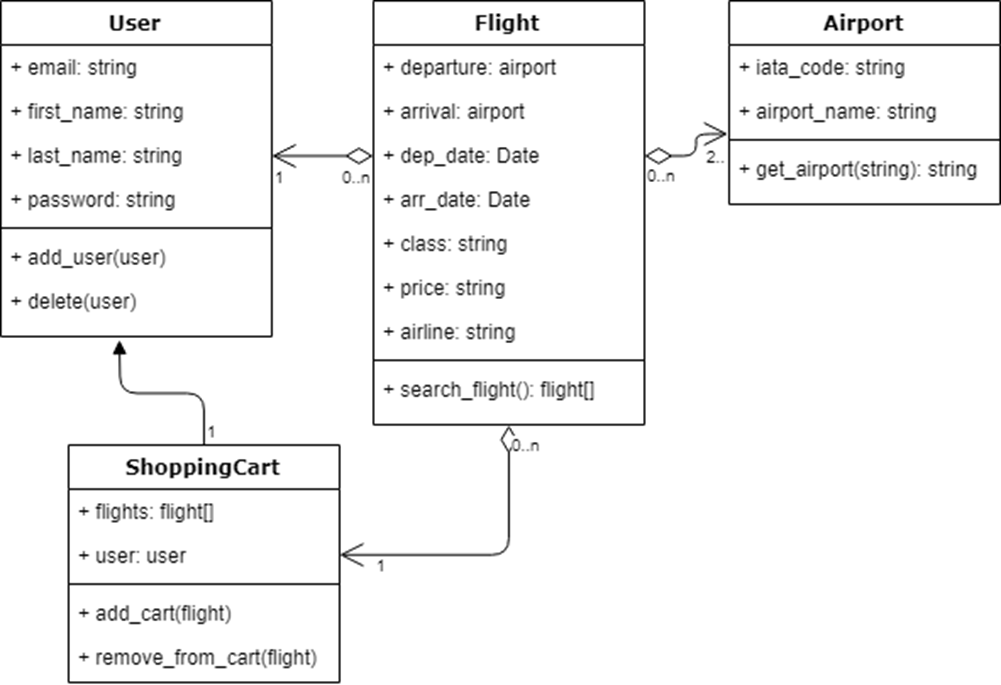
Figure Above: Microservice Architecture v2.0.

* Application can be broken down into different services for the different NDC transactions:
  + Flight Search
  + Flight Booking
  + Flight Ticketing
* Every service can be written independently with different programming languages and architecture.

Figure Above: Recommended Airline Architecture by IATA

Figure Above: Sequence Diagram, showing three main NDC transactions for ticketing a flight.

Above Figure: Use Case Diagram.

Above Figure: Class Diagram.

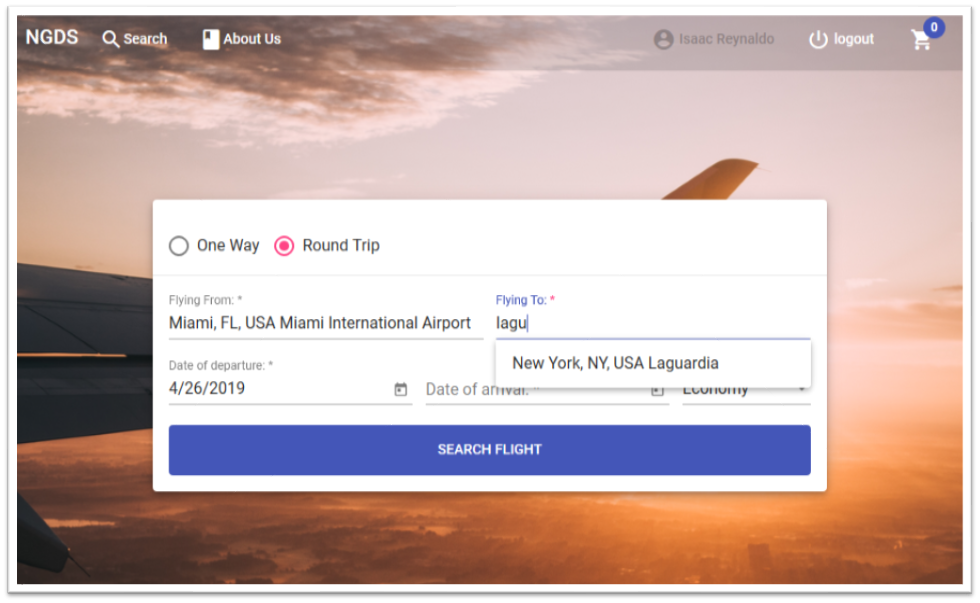
## 

## Above Figure: Interaction Diagram

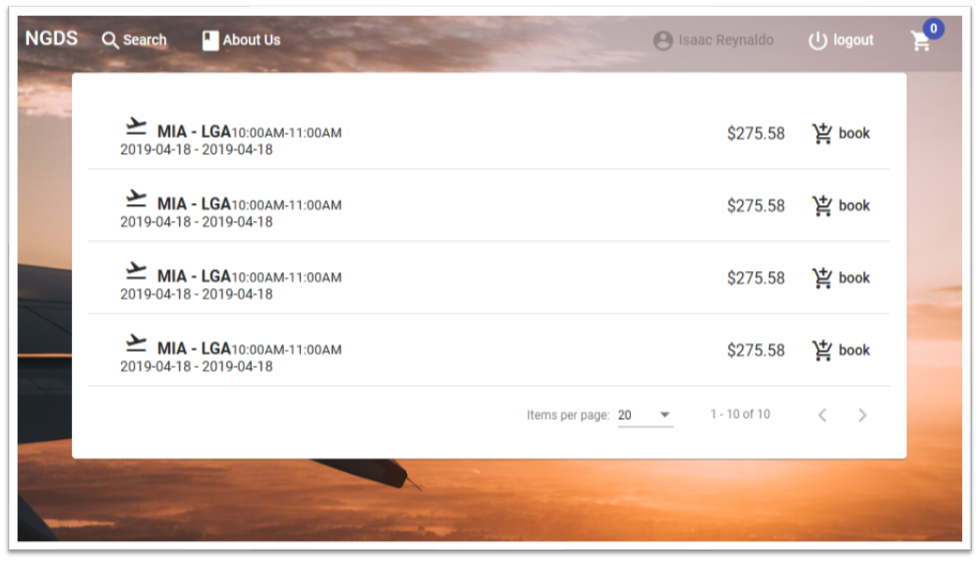
## Appendix B - User Interface Design

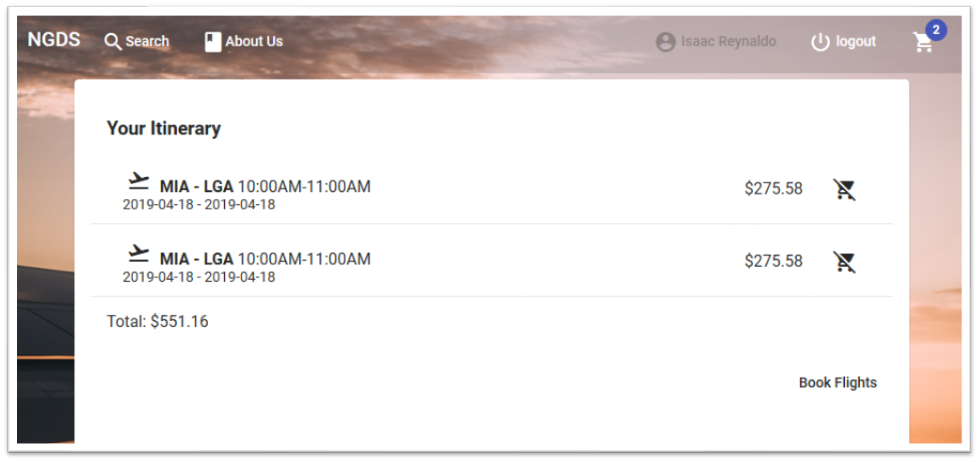
The User Interface was designed using the Angular Material Framework, following the Material Design Philosophy and Style.

Bellow are screenshots of the User Interface:



Above Picture: Searching a Flight

Above Picture: List of Available Flights.

Above Picture: Shopping Cart with Flights to Book.

## Appendix C - Sprint Review Reports

20190201 - Sprint 1 Review

Attendees: Luis Herrnsdorf, Isaac Reynaldo, Gonzalo Mera

Start time: 9:30 PM

End time: 9:50 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story:
  + NGDS-7
  + NGDS-9
  + NGDS-11
  + NGDS-10
  + NGDS-12
  + NGDS-14

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story NGDS-6
* User Story NGDS-13
* User Story NGDS-25
* User Story NGDS-16

## 

20190215 - Sprint 2 Review

Attendees: Luis Herrnsdorf, Isaac Reynaldo, Gonzalo Mera

Start time: 9:30 PM

End time: 9:50 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story NGDS-32 as a user I would like to sign up
* User Story NGDS-33 as a user I would like to log in
* User Story NGDS-26 as an airline I would like to receive shopping request.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story NGDS-30 As an Online Travel Agent I would like to Add flights to my Shopping Cart
* User Story NGDS-34 As an Aggregate Server I would like to send a flight request
* User Story NGDS-35 As an Aggregate Server I would like to keep stored recent searches
* User Story NGDS-27 As an Airline I would like to store my flights
* User Story NGDS-31 As an Online Travel Agent I would like to Delete Flights from my Shopping Cart
* User Story NGDS-29 As an Online Travel Agent or User I would like to send a query to an Aggregate Server
* User Story NGDS-28 As an Airline I would like to find flights that have been requested
* User Story NGDS-36 As an Aggregate Server I would like to listen for OTA's
* User Story NGDS-37 As an Aggregate Server I want to be able to convert OTA's request to NDC

20190301 - Sprint 3 Review

Attendees: Isaac Reynaldo, Gonzalo Mera, Luis Herrnsdorf

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story NGDS-39 As systems architect I want to identify the main components of the different systems so that I can draw a high level diagram of the app.
* User Story NGDS-29 As an Online Travel Agent or User I would like to send a query to an Aggregate Server so I can find multiple flights with my input.
* User Story NGDS-31 As an Online Travel Agent I would like to Delete Flights from my Shopping Cart so that I don't book them.
* User Story NGDS-30 As an Online Travel Agent I would like to Add flights to my Shopping Cart so that I can book them later.
* User Story NGDS-45 As a system architect I want to identify behavior of reads and writes for OTA's database usage.
* User Story NGDS-47 As a system architect I want to identify a strategy to address the reads and writes behavior of the OTA's database usage.
* User Story NGDS-27 As an Airline I would like to store my flights so that I can access them and sell them.
* User Story NGDS-28 As an Airline I would like to find flights that have been requested so that I can send back the flight references.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story NGDS-34 As an Aggregate Server I would like to send a flight request
* User Story NGDS-46 As an Aggregate Server I would like to keep stored recent searcher
* User Story NGDS-35 As an Aggregate Server I would like to keep stored recent searches for a determined time so that I can give query response efficiently.
* User Story NGDS-48 As a system architect I want to identify a strategy to address the reads and writes behavior of the Aggregate database usage.

20190315 - Sprint 4 Review

Attendees: Isaac Reynaldo, Gonzalo Mera, Luis Herrnsdorf

Start time: 9:30pm

End time: 10:00pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story NGDS-38 As a systems architect I want to define the transactional behavior of the different databases so I can identify an optimization strategy that I can use to design and implement my db.
* User Story NGDS-41 As a system architect I want to research best practices for architectural implementation of the OTAs so that I can find the most efficient method.
* User Story NGDS-43 As a system architect I want to research best caching efficiency methods for aggregator services.
* User Story NGDS-44 As a system architect I want to research best caching efficiency methods for OTAs's cache, so that I can find the most efficient way to reduce querying requests.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story NGDS-34 As an Aggregate Server I would like to send a flight request
* User Story NGDS-46 As an Aggregate Server I would like to keep stored recent searcher
* User Story NGDS-35 As an Aggregate Server I would like to keep stored recent searches for a determined time so that I can give query response efficiently.
* User Story NGDS-48 As a system architect I want to identify a strategy to address the reads and writes behavior of the Aggregate database usage.
* User Story NGDS-49 As an Airline I want to sell flights directly to travelers

20190329 - Sprint 5 Review

Attendees: Isaac Reynaldo, Gonzalo Mera, Luis Herrnsdorf

Start time: 9:30pm

End time: 10:00pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story NGDS-51 As a System Designer I want to have three instances of Airlines so I can test the NDC transactions.
* User Story NGDS-52 As a system architect I want my airline database to perform well with around 600 transactions per second of reads.
* User Story NGDS-53 As a system architect I want to use similar airline data for fares over the same time period and market.
* User Story NGDS-34 As an Aggregate Server I would like to send a flight request
* User Story NGDS-35 As an Aggregate Server I would like to keep stored recent searches for a determined time so that I can give query response efficiently.
* User Story NGDS-54 As a system architect I want to connect the OTA to the Airline endpoint so that I can start connecting the services.
* User Story NGDS-55 As a system architect I want to implement a Api Gateway so that I can connect the different microservices to it.
* User Story NGDS-56 As a user I want to search a flight in a quick way so that I can buy my ticket fast.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story NGDS-46 As an Aggregate Server I would like to keep stored recent searcher
* User Story NGDS-48 As a system architect I want to identify a strategy to address the reads and writes behavior of the Aggregate database usage.

20190412 - Sprint 6 Review

Attendees: Isaac Reynaldo, Gonzalo Mera, Luis Herrnsdorf

Start time: 9:30pm

End time: 10:00pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story NGDS-34 As an Aggregate Server I would like to send a flight request to multiple airlines so that I find more matches for the query.
* User Story NGDS-46 As a system architect I want to identify behavior of reads and writes for Aggregate service's database usage.
* User Story NGDS-48 As a system architect I want to identify a strategy to address the reads and writes behavior of the Aggregate database usage.
* User Story NGDS-57 As an User I want to store my current flight search so that I can quickly find it again.
* User Story NGDS-58 As a Aggregate Server I want to do cashing to flight search so that I don't have to unnecessarily look for a flight straight from the Airline.
* User Story NGDS-59 As a system Architect I want to connect to three different Airlines so that I can aggregate the correct flights being requested.
* User Story NGDS-61 As a system Architect I want to create a Poster with the process of the project so that I can finalize the Project.
* User Story NGDS-63 As a System Architect I want to create a Open Source License so that the project can be exposed to the open source community.
* User Story NGDS-64 As a system Architect I want to create documentation of the process of the project so that I can finalize the Project.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* User Story NGDS-60 As a System Architect I want to verify NDC schemas so that I know when I'm using the correct protocol, and receiving it.
* User Story NGDS-62 As a software developer I want to finish documenting the project so that I can make it easier to continue the Project

## Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents.

## User Manuals:

PostgreSQL Database manual

* https://www.postgresql.org/files/documentation/pdf/11/postgresql-11-US.pdf

NodeJs documentation

* <https://nodejs.org/dist/latest-v10.x/docs/api/>

Express Framework api

* <https://expressjs.com/en/4x/api.html>

Nginx documentation

* <https://docs.nginx.com/>

## Installation/Maintenance

Install PostgreSQL

* <https://www.postgresql.org/docs/11/tutorial-install.html>

Install NodeJs

* <https://nodejs.org/en/download/>

Install Express

* <https://expressjs.com/en/starter/installing.html>

Install Nginx

* <https://www.nginx.com/resources/wiki/start/topics/tutorials/install/>

## Shortcomings/Wishlist

* Implement API from <http://open-ndc.org/>
* Improve implementation of ancillaries in Airline web app

## References

<https://www.iata.org/whatwedo/airline-distribution/ndc/Pages/default.aspx>

<http://open-ndc.org/>