PizzaDronz Reflection Essay

Part 1: Implementation experiences

As the first step of app development, requirements are set up by capturing needed features and other important factors such as market trends [3]. Even though the requirements are always scoped to plan the implementation, design, and structure of the app, it must be considered that there are unforeseen changes in requirements, and so does PizzaDronz app development.

Changes in requirements

The requirements set in the stage of project planning were mostly kept the same as they guided the implementation of the app. However, it is always inevitable to keep the entire plan the same due to some factors.

Functional Requirements

- For validity check on order information, LocalDate was planned to be used to find if the provided expiry date of the credit card has been passed or not. However, as the function checks the validation of the order, the credit card's expiry date is compared with the date that the order was made.
- A* algorithm is used to find the optimal path between Appleton Tower and the restaurant.
- To lessen the amount of time it takes for the program to run, path from each restaurant to Appleton Tower is calculated and stored in a map. The paths for different restaurant will be used to generate a path for an order by reversing the path from restaurant to Appleton tower.

Non-functional Requirements

- Regarding security, it was planned to avoid the defined no-fly zone for the drone to fly over. However, in the case of two associated positions outside no-fly zone, the line connecting the two points cuts off the no-fly zone. As the algorithm explores nodes that has the lowest cost to find the optimal path, the system allows such cases.

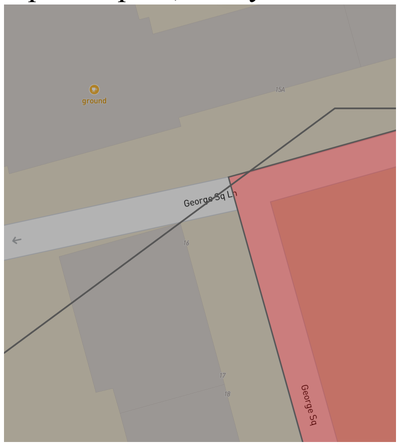

Figure 1 Association of two positions slightly cuts off the edge of no- fly zone.

Implementations

The primary purpose of developing PizzaDronz is to use a drone to deliver pizza from restaurants with the shortest path, avoiding defined no-fly zones.

Data

To obtain real-time data about orders information, restaurants information, central area and no-fly zone information, and if the server is alive. The REST-service provides data in JSON format. The related data were used and implemented in many different data structures, mainly Maps. To convert the JSON data suitable to the app development, an object mapper was used. This enabled transformations of object between classes and App.java to generate output files with result of calculations on valid flight paths and update on order status.

Flightpath-finding algorithm

As one of the main functions of this app, flight path algorithm calculates the shortest path between two points. For the implementation of the function in the app, a* algorithm was used. The reason of selecting a* algorithm is that it engages the completeness of the solution found, comes up with the optimal path with the least cost, and uses heuristics that make the algorithm visit fewer nodes [1]. 'LngLat' positions are used as nodes that a* algorithm explores. To make sure the algorithm is efficient, the PriorityQueue data structure was used to store positions in an open set based on their f scores.

App.java

In App.java, all the data were retrieved from REST server. Using the obtained data, validation check on orders are done before calculating flightpath, and the flightpaths are mapped with corresponding orders. The results are put into output files generated with valid base URL and date provided.

Reflection

With some modification to the original plan for development of the program, the application has been implemented with a clear concept of efficiency, optimality, and completeness. However, there are some possible improvements to usablilibty and efficiency of implementing the application.

The application is developed in a monolithic architecture, retrieving all data from one REST server database. The app has different usage of data – order, restaurant, central area, and no-fly zone. Due to this configuration, the implementation of the application is complicated. The hardest part of developing this program was that all modules and components are interconnected, which made it difficult to identify and mitigate problems. A suggestion for this limitation of the application development is to use micro-service architecture. In this architecture, components are grouped for different purposes and usage. This enables independent test that will make it much easier and faster to point out the issue of a certain class or function and make changes.

Part 2: PizzaDronz Mobile Application Architecture

For a mobile application to perform tasks efficiently, decisions made for good architecture are important. An application with good architecture consists of independent components, so called modularity [2]. This is the key to modification of the app within less time and cost. Moreover, developing a mobile application that is constructed in a good architecture saves time and cost for other projects by allowing reuse of code. The architecture contributes to the improvement of security in an app by breaking up sensitive sources and engaging security protocols around it.

Types of mobile app architectures

A mobile app is developed in an architecture consisting of multiple layers. Depending on the type of the mobile app architecture, the structures and layers alter. There are three common types of mobile app architecture: Layered architecture, Monolithic architecture, and Microservices architecture.

The layered architecture consists of layers or tiers where components are organized in several groups of logic and purposes. On-premises software, where software solutions are operated only on an organization's premises, is a typical application of this type of architecture. Layers share data only with an adjacent layer. In this regard, managing the dependencies and security of the app is more effective, but there is a limitation that it is harder to find an error from a specific layer in larger apps. [2]

In a monolithic architecture, the app has a single unit called monolith where all components function, firmly integrated. The main feature of such architecture is its simplicity due to the strong integration within the app. As a self-contained app, developers often find it easier to create, test, and deploy the app. However, the simplicity is also a disadvantage of monolithic architecture as modification and update of the app are extremely limited in the aspects of time and cost. [2]

While the first two architectures are not recommended for the comprehensiveness and flexibility of developing a mobile app, microservices architecture is an advantageous approach for the app development. Microservice is where the app is split into smaller components, which communicate with the main app through an application programming interface (API). [2] The reason for the high flexibility and fluidity of this approach is that it allows independent development, tests, and deployment of each function in the app. This is the best approach to be taken for complicated app development. The food delivery mobile app is where microservice architecture is the most suitable. Here is an example, Uber – transportation and food delivery service mobile app, that has switched the architecture from monolithic to microservice [11]. Here is a diagram of PizzaDronz mobile app in microservice architecture:

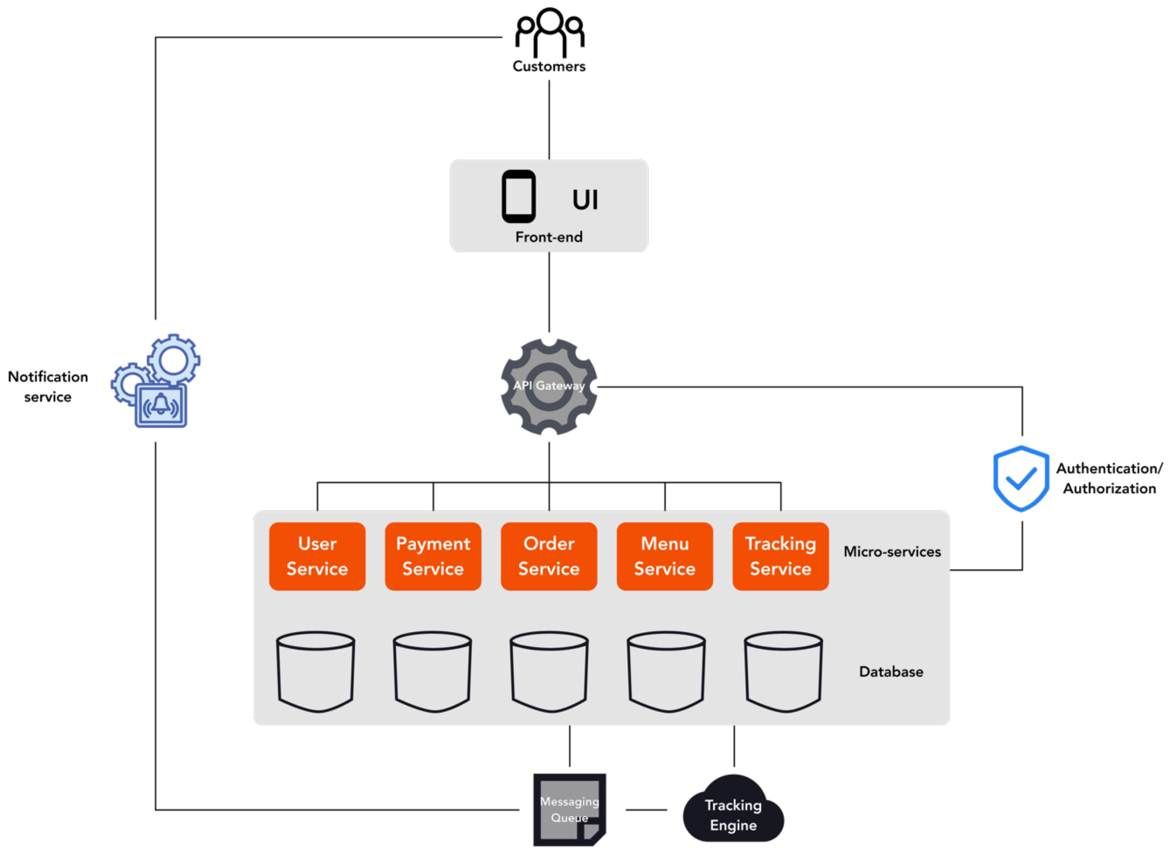

Figure 2 Micro-service architecture

External architecture

Here is a brief overview of the external architecture of the mobile app. This is the client-facing part of the mobile app where customers use to order, browse the menu, and track delivery status [8]. The core functions focused are listing restaurants, viewing menu, placing order, payment method, and order tracking. Any requests the user makes through the front-end part of the app are transmitted to the back-end structure, or internal architecture, of the app to send back the right response. The design is as below:

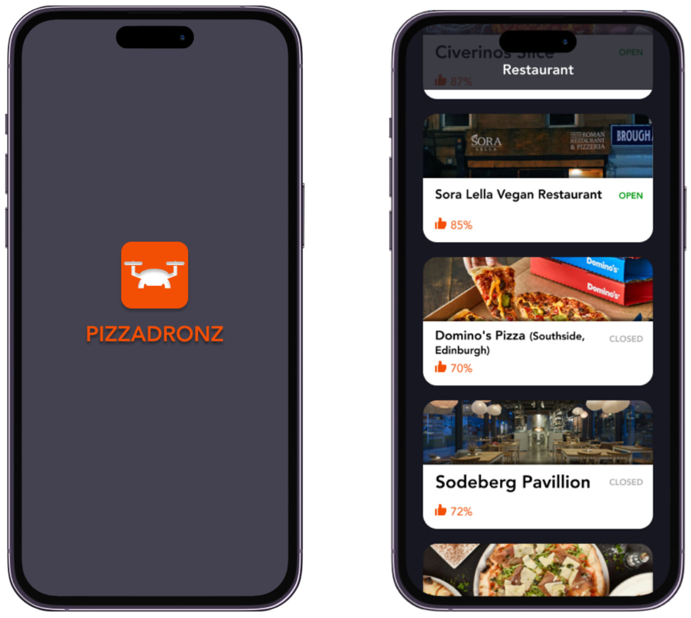

Figure 3 Starting Page of PizzaDronz Mobile App (left); PizzaDronz restaurant list (right)

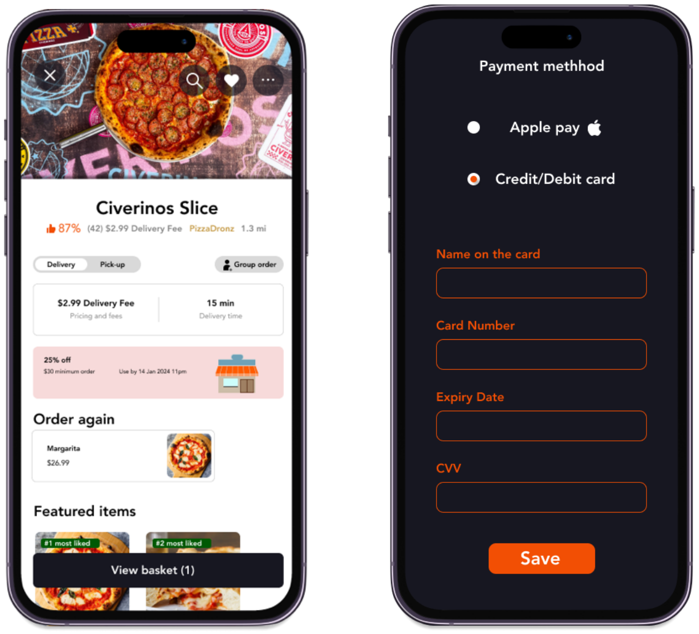

Figure 4 Page of menu in a restaurant (left); payment method insert form (right)

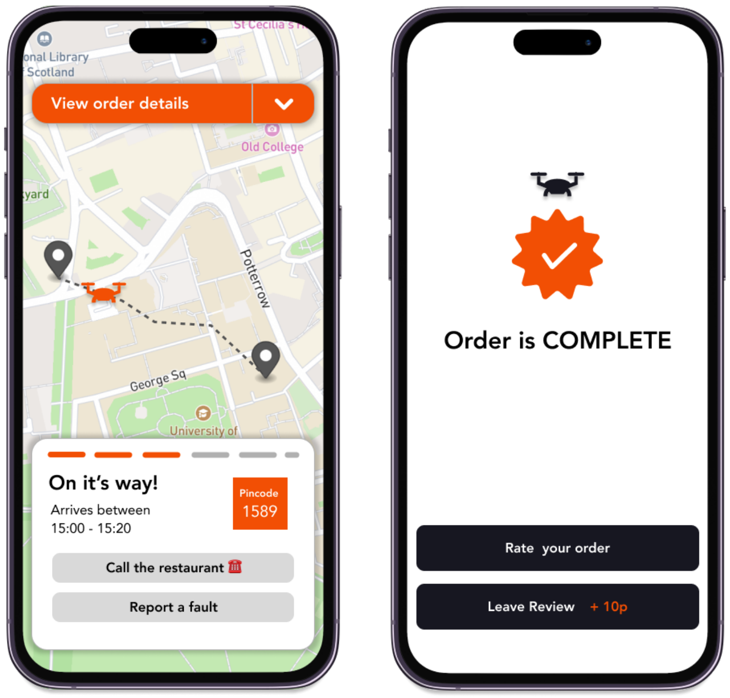

Figure 5 Order tracking screen(left) and Notification for completed order with order rating and review options (right)

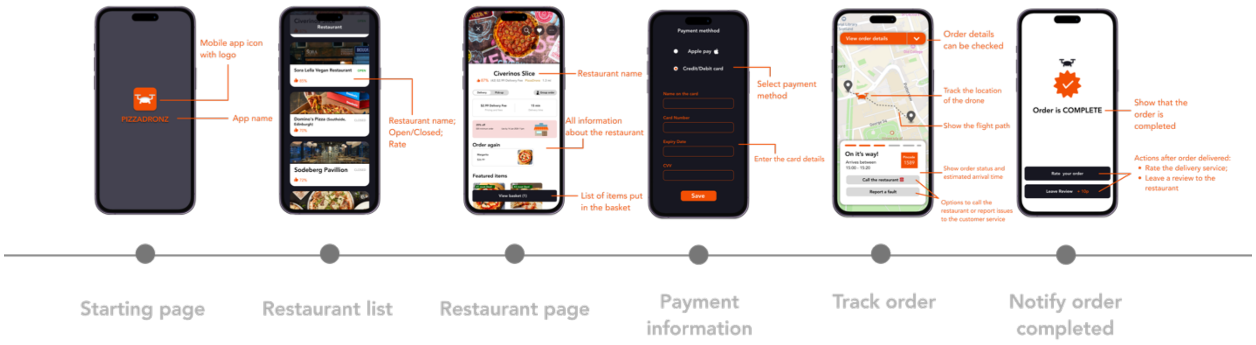
Mobile app icon with logo
PIZZADRONZ
App name
Restaurant name; Open/Closed; Rate
Civerinos Slice
Restaurant name
All information about the restaurant
List of items put in the basket
Payment methhod
Apple pay
Credit/Debit card
Select payment method
Enter the card details
Save
Order details can be checked
Track the location of the drone
Show the flight path
Show order status and estimated arrival time
Options to call the restaurant or report issues to the customer service
Order is COMPLETE
Show that the order is completed
Actions after order delivered:
Rate the delivery service;
Leave a review to the restaurant
Starting page
Restaurant list
Restaurant page
Payment information
Track order
Notify order completed

Figure 6 High Fidelity Wireframe of PizzaDronz mobile app

Internal Architecture

In micro-services architecture applied to PizzaDronz app, there are microservices of user, order, payment, tracking, and menu (restaurant) services. Each microservice has different databases, categorized into two main types of databases: transactional databases and JSON documents.

Transactional databases

Transactional database stores data as rows so that import and export of data is more readily done [6]. The database is also relational for high scalability when the service manages concurrent orders and users. The services that need such databases are user, order, and payment services [8].

JSON document storage

JSON databases allow more flexible and scalable storage of data by managing data partitioning, indexing, clustering, replication, and data access [5]. This enables more approachable and scalable search options. For this reason, information about restaurants, menus, prices, and offers is managed in JSON document storage.

These services, while they have their own databases in different types, are connected through an API gateway, where the application flow is controlled mainly in this mobile app. Requests sent by the customers are routed to the appropriate services. In other words, API gateway plays the most important role of communication within the application [2].

Here is an overview on communication between app components and the client-facing part of the app, transforming the information in the microservice architecture:

When the customer place orders from the app, the front-end architecture will send the information to the messaging queue. This will get the needed information from the microservices corresponding to the data they require. Restaurant will be informed about the order and the drone will be configured with the flight path to pick up and deliver the order. This process includes the collection of data from different microservices, transforming the integration of information through API gateway back to the client-facing part.

Conclusion

As a complicated application, microservice architecture is an optimal approach for its higher scalability, efficiency, and flexibility for PizzaDronz. While the deployment of the application in such architecture requires more time and effort as the application gets larger, it saves much more dedication to the development, modification, and upgrade of the application. This is because it is easier to identify issues and make needed changes and maintenance to the specific part of the app. The microservice architecture also benefits to the modularity. This means that app distributed into components can be developed, modified, and tested independently. While there is a disadvantageous aspect of this architecture, the benefits balance the possible issues to the more than sufficient extent.

Reference list

[1] A S, R. (2021). *A* Algorithm in Artificial Intelligence You Must Know in 2023 | Simplilearn*. [online] Simplilearn.com. Available at: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm#:~:text=The%20A.

[2] Agency, D. (2023). *The mobile app architecture guide*. [online] decode.agency. Available at: https://decode.agency/article/mobile-app-architecture/#:~:text=in%20the%20process.-.

[3] Jain, P. (n.d.). *How Long does It take to Build an App*. [online] Startup Grind. Available at: https://www.startupgrind.com/blog/how-long-does-it-take-to-build-an-app/#:~:text=For%20the%20smaller%20version%2C%20it.

[4] Mitrofanskiy, K. (2023). *Developing an Uber-like app: the tech stack and software architecture*. [online] Intellisoft. Available at: https://intellisoft.io/the-foundation-of-uber-the-tech-stack-and-software-architecture/#:~:text=The%20microservice%20architecture%20allows%20the [Accessed 30 Nov. 2023].

[5] MongoDB. (n.d.). *What Is A JSON Database? | All You Need To Know*. [online] Available at: https://www.mongodb.com/databases/json-database.

[6] Prad, R. (2021). *Introduction to Layered Architecture for Microservices*. [online] www.sayonetech.com. Available at: https://www.sayonetech.com/blog/layered-microservices-architecture/#:~:text=A%20layered%20microservices%20architecture%20is [Accessed 30 Nov. 2023].

[7] Schmuck, D. (2022). *On-Premise: How does the server-based software model work?* [online] TeamDrive - Secure Collaboration. Available at: https://teamdrive.com/en/blog-en/on-premise-server-based-software/# [Accessed 30 Nov. 2023].

[8] Sharma, P. (2023). *Architecture and Design Principle for Online Food Delivery System*. [online] pratapsharma.io. Available at: https://pratapsharma.com.np/architecture-of-food-delivery-app [Accessed 30 Nov. 2023].

[9] Soma (2023). *What is Database Per Microservices Pattern? What Problem does it solve?* [online] Javarevisited. Available at: https://medium.com/javarevisited/what-is-database-per-microservices-pattern-what-problem-does-it-solve-

60b8c5478825#:~:text=The%20Database%20per%20Microservice%20design%20pattern%20solves%20the%20problem%20of [Accessed 30 Nov. 2023].

[10] www.ibm.com. (n.d.). *What is a relational database? | IBM*. [online] Available at: https://www.ibm.com/topics/relational-databases#:~:text=Relational%20databases%20are%20transactional [Accessed 30 Nov. 2023].

[11] Yadav, M. (2023). *Understanding Food Delivery App Architecture and Functionality*. [online] ValueAppz - Blog. Available at: https://www.valueappz.com/blog/food-delivery-app-architecture-and-functionality.