

UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Freight trs: freight transportation management app (carrier flow)

Author:
Oleh TYZHAI

Supervisor:
Serj MISKIV

*A thesis submitted in fulfillment of the requirements
for the degree of Bachelor of Science*

in the

Department of Computer Sciences
Faculty of Applied Sciences



APPLIED
SCIENCES
FACULTY ●

Declaration of Authorship

I, Oleh TYZHAI, declare that this thesis titled, "Freight trs: freight transportation management app (carrier flow)" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

“When something is important enough, you do it even if the odds are not in your favor.”

Elon Musk

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Freight trs: freight transportation management app (carrier flow)

by Oleh TYZHAI

Abstract

Many people have no idea how freight affects their daily lives and how important it is for these processes to be perfectly tuned and automated, thinking that it is too global and does not apply. But in fact, due to transportation, people are provided with the products of their daily use because the end consumer for any product is a person. If the truck had not brought fuel to the gas station, the person would not have been able to refuel. If the truck did not deliver food to the supermarket, the person would not be able to buy it. If the truck hadn't delivered the wood to the furniture factory, people wouldn't have furniture.

Acknowledgements

First of all, I would like to thank my manager Serge Miskiv for contributing to the project. I also want to thank everyone involved in the transport industry who inspired me to come up with the idea for this project. Thank you to all the teachers of the university for giving me enough knowledge to implement the project. And special thanks to parents and loved ones for their faith and daily support.

Contents

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
1 Introduction	1
1.1 Motivation	1
2 Existing solutions	2
2.1 Uber Freight	2
2.1.1 Overview	2
2.2 Della	3
2.2.1 Overview	3
3 Proposed approach	4
3.1 Overview	4
3.2 Architecture	4
3.2.1 SwiftUI + Redux	4
3.2.2 Model - View - ViewModel	5
3.2.3 Model - View - Controller	5
3.2.4 Summary	6
3.3 UI/UX design	7
3.4 Flows: Client, Carrier - Driver and Manager	10
3.4.1 Carrier	10
Main features:	11
Main features:	11
3.4.2 Client	11
3.5 Libraries and frameworks	13
3.5.1 Google Maps	13
3.5.2 Stripe	13
3.5.3 Twilio	13
3.6 Backend	13
3.6.1 Overview	13
3.6.2 Django	14
3.6.3 Rest Framework	14
All user types	14
Managers only	14
Drivers only	14
Customers only	15
3.6.4 Authentication	15
3.6.5 Heroku	15
3.6.6 PostgreSQL	16

4	Future plans	17
4.0.1	Documentation	17
4.0.2	Contacts feature	17
4.0.3	Carrier statistics	17
4.0.4	Truck integration	17
5	Summary	18

List of Figures

3.1	Redux flow	4
3.2	MVVM flow	5
3.3	MVC flow	6
3.4	Order lifecycle	12
3.5	Postico app	16

List of Tables

2.1	Pros and Cons of Uber Freight	2
2.2	Pros and Cons of Della	3
3.1	Pros and Cons of Redux	4
3.2	Pros and Cons of MVVM	5
3.3	Pros and Cons of MVC	6

List of Abbreviations

LAH	List Abbreviations Here
WSF	What (it) Stands For
IT	Information Technology
MVVM	Model View View-Model
MVC	Model View
UCU	Ukrainian Catholic University

Dedicated to my lovely parents

Chapter 1

Introduction

1.1 Motivation

Working in the agriculture area with my parents, I communicated with hundreds of truck drivers, carrier agents, drivers who have their own trucks, and I noticed that Ukraine has a huge problem in the transportation area. Many truck owners were forced to sell their business and trucks just because they could not find proper orders and provide tasks for their employees, or trucks spent more time without cargo. For example, it is easy to find any order from Lviv to the port of Odesa, but it is tough to find any orders in the opposite direction. Even at school age, I began to analyze this problem in more depth and realized no products on the market that completely solve these problems. After talking to various companies in need of transportation and companies involved in transportation, I came up with a system that could fully automate the transportation process, minimize the percentage of distance traveled without cargo, thereby reducing the number of harmful gases emitted by trucks into the atmosphere and increase profits for carriers.

But at that time, my ideas were just a dream because I was just a schoolboy who had no experience in IT. That's why I decided to enter the computer science program at UCU and at the end of my studies to get closer to my goal.

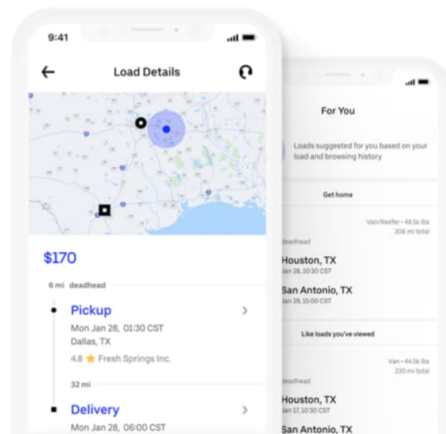
Chapter 2

Existing solutions

2.1 Uber Freight

2.1.1 Overview

In 2018, the American company Uber introduced its new system called Uber Freight. The basic idea of this system is very similar to mine. I read several articles about uber freight transport and read the comments of real users of this system and saw a lot of hatred. Many people say that the application is terrible and has many problems with order fulfillment and payment. A lot of bad feedback comes from drivers and dispatchers. It can conclude that Uber has focused more on the customer side of the program. It is not surprising because it is the customer who makes the profit, and if there are a large number of orders in the system, the carriers join automatically. From a business standpoint, this is probably the right decision, but in my app, I would like to place a strong emphasis on the carrier app to encourage them to import orders even out of the system.



Pros	Cons
Modern	not available in Ukraine
User-friendly	focused on customers

TABLE 2.1: Pros and Cons of Uber Freight

2.2 Della

2.2.1 Overview

Della is the only platform available on the Ukrainian market, but in my opinion, it is pretty simple and does not solve all the problems associated with transportation. This is essentially a bulletin board where customers place orders, leave their phone numbers, and wait for carriers to call them. On the one hand, this is a pretty good solution because the development of such a system does not take much time, and at the moment, it is actively working, and users are satisfied. But in my opinion, this system is outdated, and developers need to focus on improvement every year to make people's lives easier.

The screenshot displays the Della website interface. At the top, there is a header with the Della logo, a date 'Четверг 06 мая 2021 г.', and language selection options for 'Украина' and 'рус'. Below the header, there are navigation buttons for 'Вход' and 'Регистрация'. The main content area is divided into several sections:

- Грузы для перевозки Украина - Украина**: A list of transport orders with columns for date, route, cargo type, weight, and volume.

Дата	Маршрут	Груз	Вес	Объем
06.05 00:27	Червоноград (UA) — Северодонецк (UA)	продукты	0,6 т	26 м³
06.05 00:26	Нижняя Ольховая (UA) — Мариуполь (UA)	брус в пачках	22 т	38 м³
06.05—07.05 00:17	Николаев (UA) — Днепро (UA)	черепица на пал...	4,5 т	
- Грузоперевозки Украина. Статистика цен**: A line graph showing price trends for 20-ton trucks from May to April. The y-axis represents price in UAH per km, ranging from 16 to 24. The x-axis shows months from May to April. A solid green line represents the price change, and a dashed red line represents the trend.
- Курсы валют Украина**: A section showing exchange rates for USD (27,79 грн) and EUR (33,34 грн).
- Поиск попутных грузов**: A search form with dropdown menus for 'Откуда' (Ukraine) and 'Куда' (Poland).
- Поиск попутных машин**: A search form with dropdown menus for 'Откуда' (Germany) and 'Куда' (Ukraine).
- Расчет расстояний**: A button for distance calculation.

Pros

works in Ukraine
easy to use

Cons

not automated
no safe payments
no documentation
no tracking

TABLE 2.2: Pros and Cons of Della

Chapter 3

Proposed approach

3.1 Overview

The proposed approach is to implement a program that can fully automate the transportation process from route planning, preparation of documents for execution, and payment

3.2 Architecture

It was pretty tricky to find the perfect architecture for the project, and before settling on the MVC, I went through many different architectures.

3.2.1 SwiftUI + Redux

It is an entirely new architecture and technology, which is practically not used in commercial projects but is quite interesting to study. Redux is a state-based architecture that requires only three elements: enum of states, enum of actions, and reducer function that take on the input the current state of the app and the action and return the new state.

Pros	Cons
Modularity	lack of sufficient documents
Easy to integrate new features	takes time to study
Easy to set up	
Easy to check	

TABLE 3.1: Pros and Cons of Redux

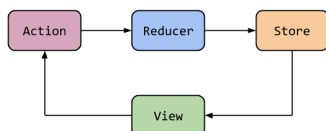


FIGURE 3.1: Redux flow



FIGURE 3.2: MVVM flow

3.2.2 Model - View - ViewModel

MVVM (Model - View - ViewModel) is an architecture template used to separate business logic from user interface logic. The main purpose of this template is to unload the inspection controller and follow the principle of sole responsibility from the SOLID template.

According to the mvvm template, the view controller must contain only the logic associated with updating information on the interface and for receiving signals from the user.

The model is used to interact with third-party services and the server database to obtain data displayed in the interface. The view model is used as an adapter between the view and the model in order to convert the data provided by the model to those that can display the view and vice versa

Pros	Cons
modularity easy to test easy to support	needs much time to integrate (there are projects for which setting up the mvvm architecture takes more time than writing a complete application using MVC)

TABLE 3.2: Pros and Cons of MVVM

3.2.3 Model - View - Controller

MVC is a fairly old and simple architecture used by thousands of applications and is used by default in iOS applications. It is an architecture that divides people into two categories. Some say that this is not the MVC that was in the '70s and that those people who consider it bad do not really know how to use it properly. Others say that using MVC is not suitable for large projects, and this architecture violates all possible programming principles. I belong to this category of people, and I can confidently say that the more the project grows, the harder it becomes to support it, add new features, or even attract new developers. Despite all this, it is ideal for small projects, and it is pretty easy to rewrite when the project starts to grow.

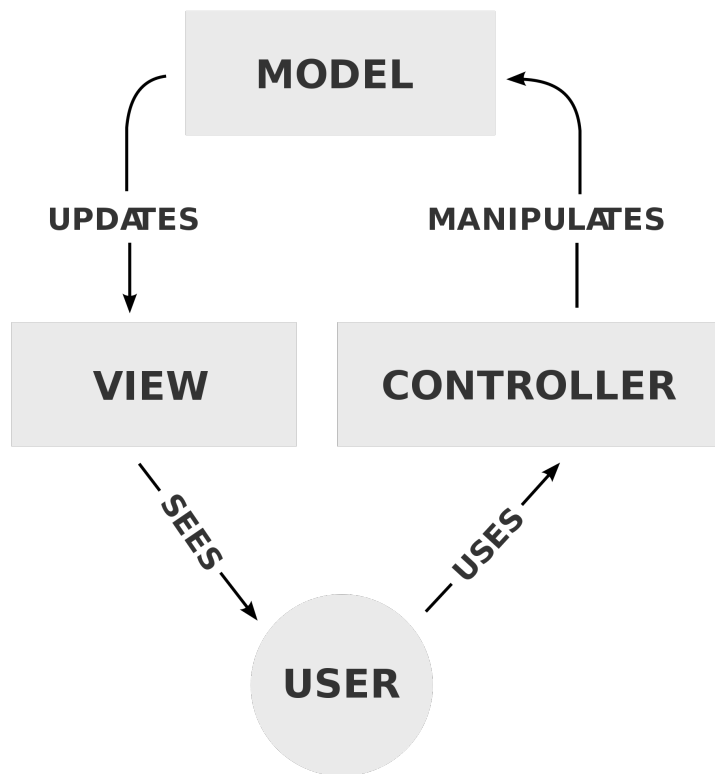


FIGURE 3.3: MVC flow

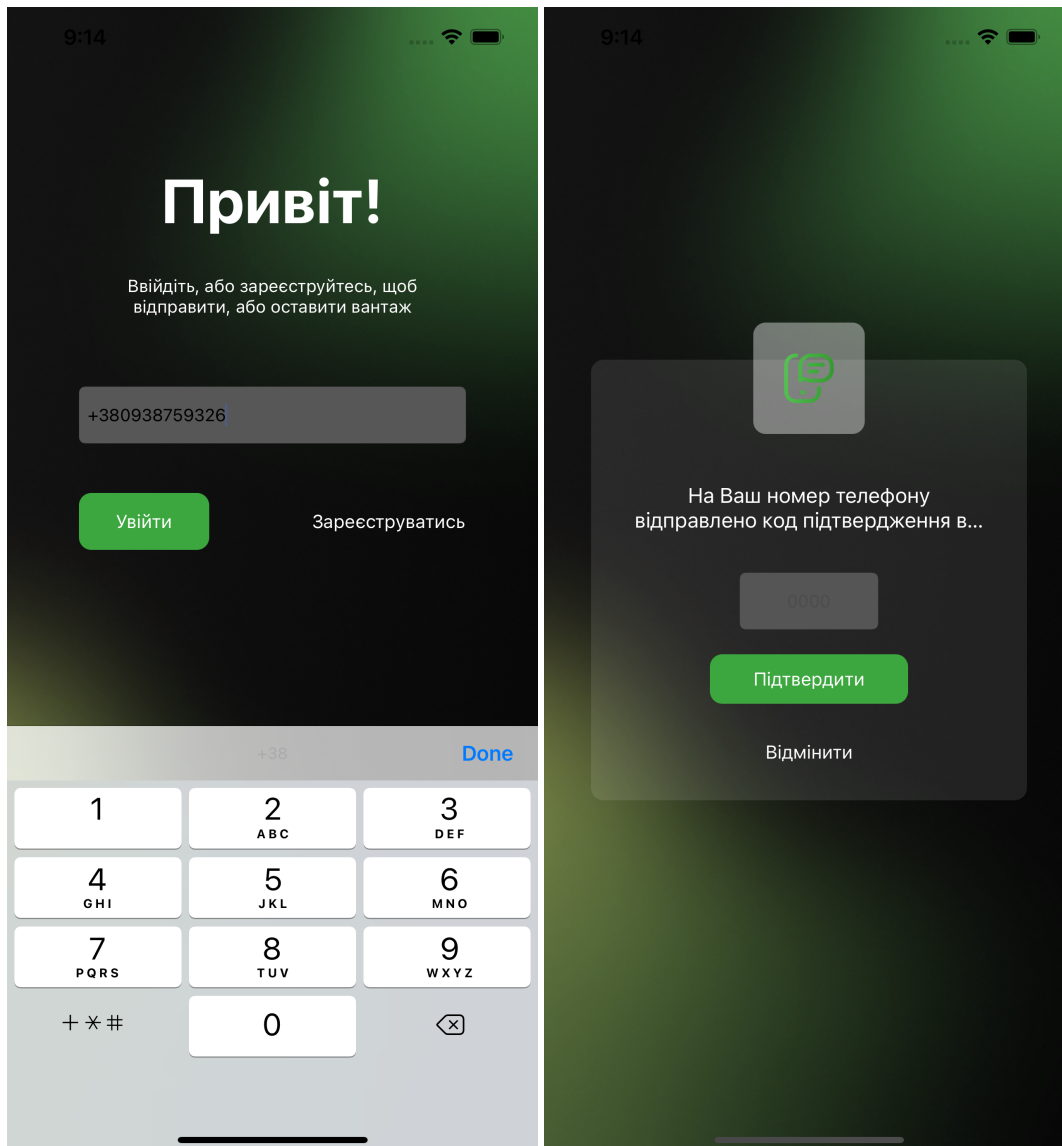
Pros	Cons
Fast in coding	hard to support unreadable hard to debug hard to test

TABLE 3.3: Pros and Cons of MVC

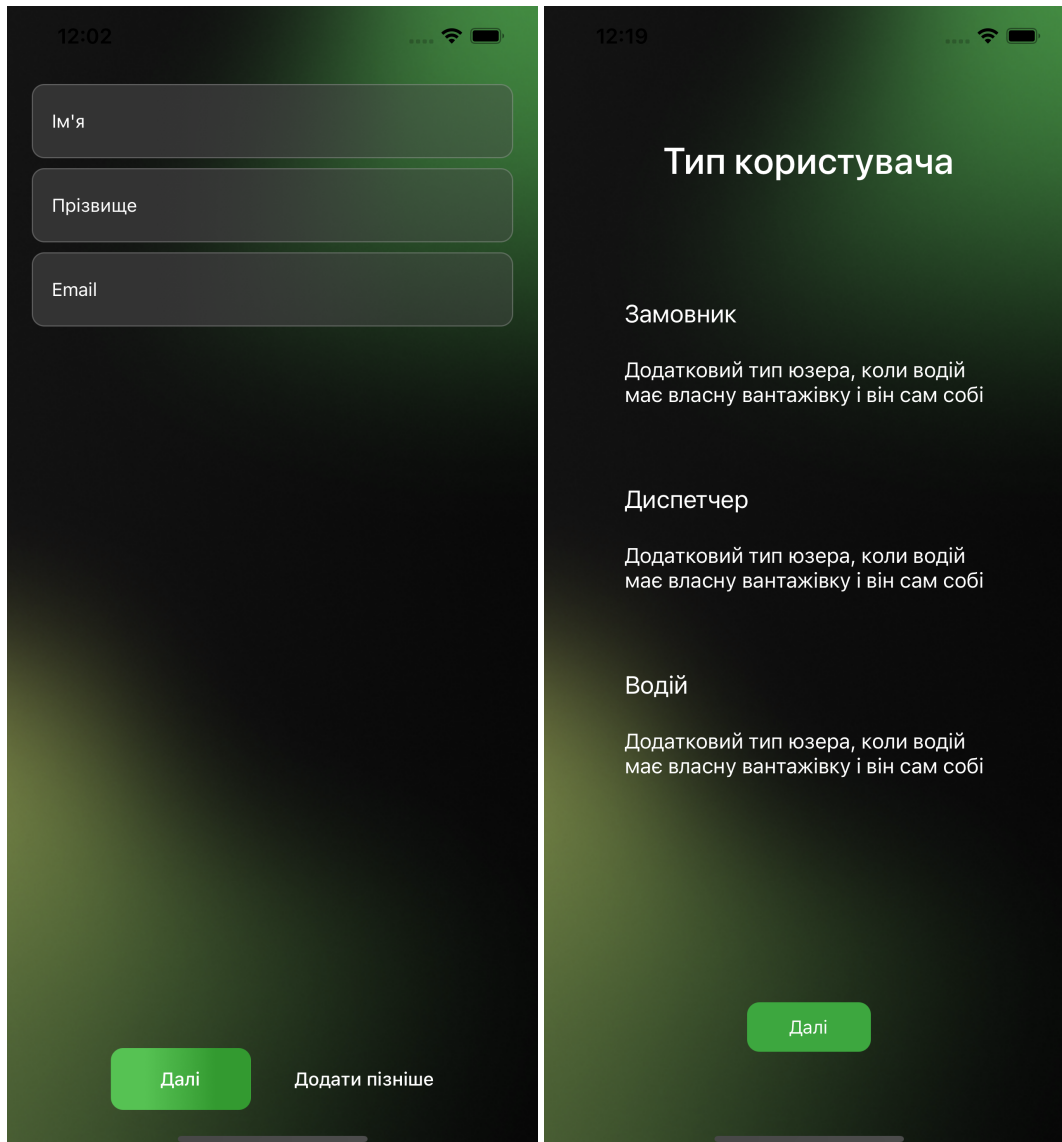
3.2.4 Summary

After weighing all the pros and cons, I still decided to use MVC in my project because for a start, I would like to make an MVP project, and since I did not have enough time for the project, MVC would be perfect for it.

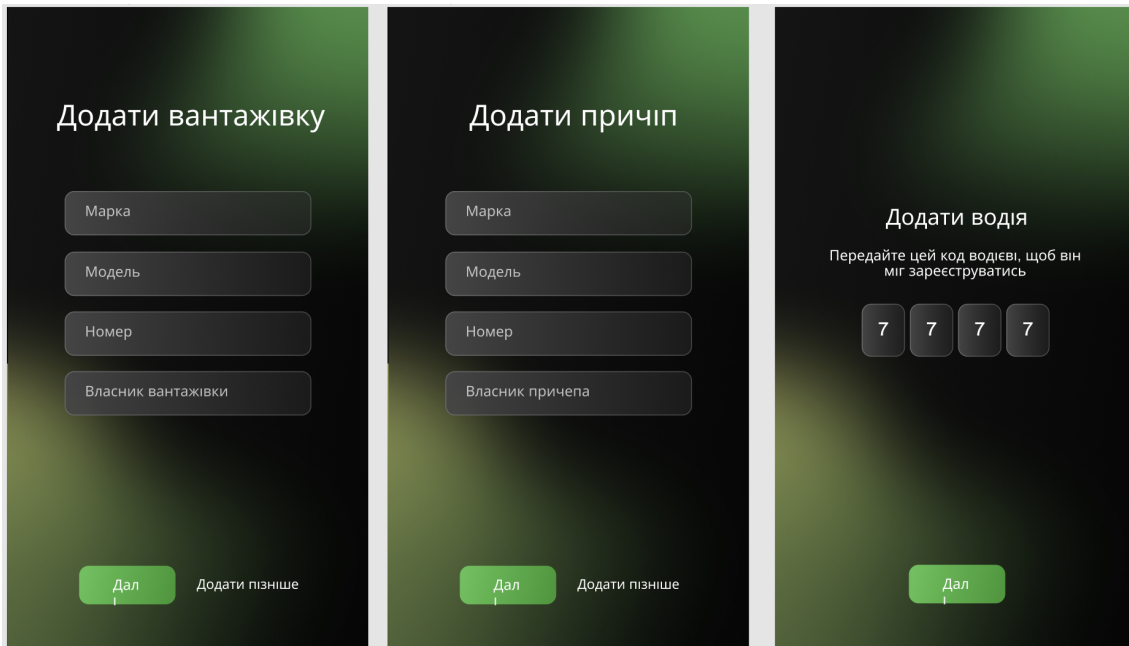
3.3 UI/UX design



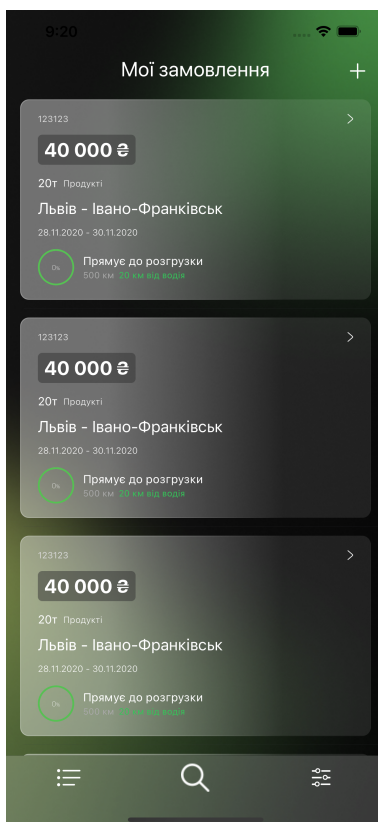
In the application, we have authorization using SMS code confirmation. On the first screen, we have a start page where the user must enter his phone number, and after pressing the login button, SMS with code will be sent to this phone number, and the user will go on the code confirmation screen. After the user submits the valid confirmation code, the app will send the request to the server to log in with the OTP code and get a response with access and refresh tokens. If the user type is selected for the current user, the app will automatically navigate the related flow. If not, then the app will guide the user to another flow to complete registration.



If the user is not yet registered in the system, the application will ask him to fill in the basic information and select the user type

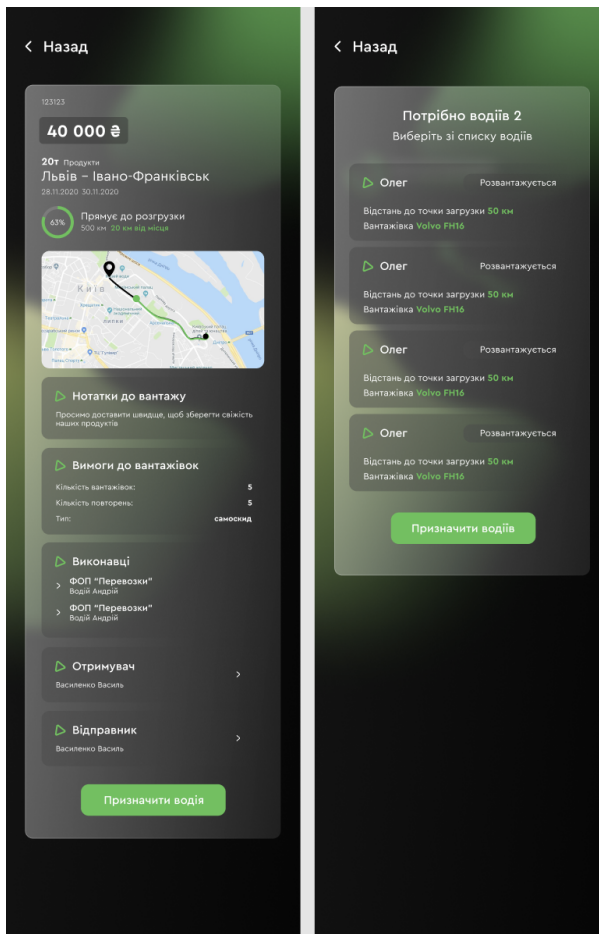


If the manager has not yet added any trucks, trailers or drivers at startup, the application will ask the user to add this data or skip this step and add it later from the settings page



On the main page, the dispatcher can see all his current jobs with their statuses and locations, and also on another tab, he can discover new orders and assign it to

his drivers



On the screen of order details, the dispatcher can see all the information about the work, read customer notes, see information about the origin and destination, their rating, find out information about the parameters of the cargo, its weight, the number of required trucks. If the dispatcher is suitable for this order, he can accept it, and his next step will be to choose a driver or several drivers who will fulfill it. When selecting a driver, the dispatcher can see what status the driver is and how far he is from the current cargo.

3.4 Flows: Client, Carrier - Driver and Manager

Since the system is quite large and includes many features, I decided to divide it into two parts: Carrier and Client

3.4.1 Carrier

- **Dispatcher.** The dispatcher acts as an intermediary between the customer and his direct executor - the driver. The essence of the dispatcher's application is to manage vast fleets of trucks and provide work for all. The manager can see all orders available in the database and not yet commissioned. Can sort or filter

them by various parameters (for example, by distance from the driver to the place of loading, by type of cargo, by weight, or by distance)

Main features:

- search for orders that are in the database and have not yet been commissioned
 - sorting or filtering them by various parameters (for example, by distance from the driver to the place of loading, by type of cargo, by weight or by distance)
 - review and accept work orders, assign tasks to the driver
 - track the process of order fulfillment and track the placement of the truck on the map in real time
 - view mileage statistics with or without cargo, see the company's profits as a whole and for each truck
- Driver The driver application is installed directly in the truck on the driver's mobile phone or tablet.

Main features:

- see the list of tasks to be performed
- accept a job order or reject it
- get the route from the current place to the place of departure and from the place of departure to the destination
- give an assessment and comments on the origin and destination
- communicate with customers and recipients via chat
- view the number of simplest kilometers in recent times
- complete the task

3.4.2 Client

This program flow is intended for individuals or companies who want to order delivery

- create an order and add it to our database
- it is safe to pay for the order
- track the location of the truck in real time
- give feedback for the driver and the carrier
- get the best price for transportation

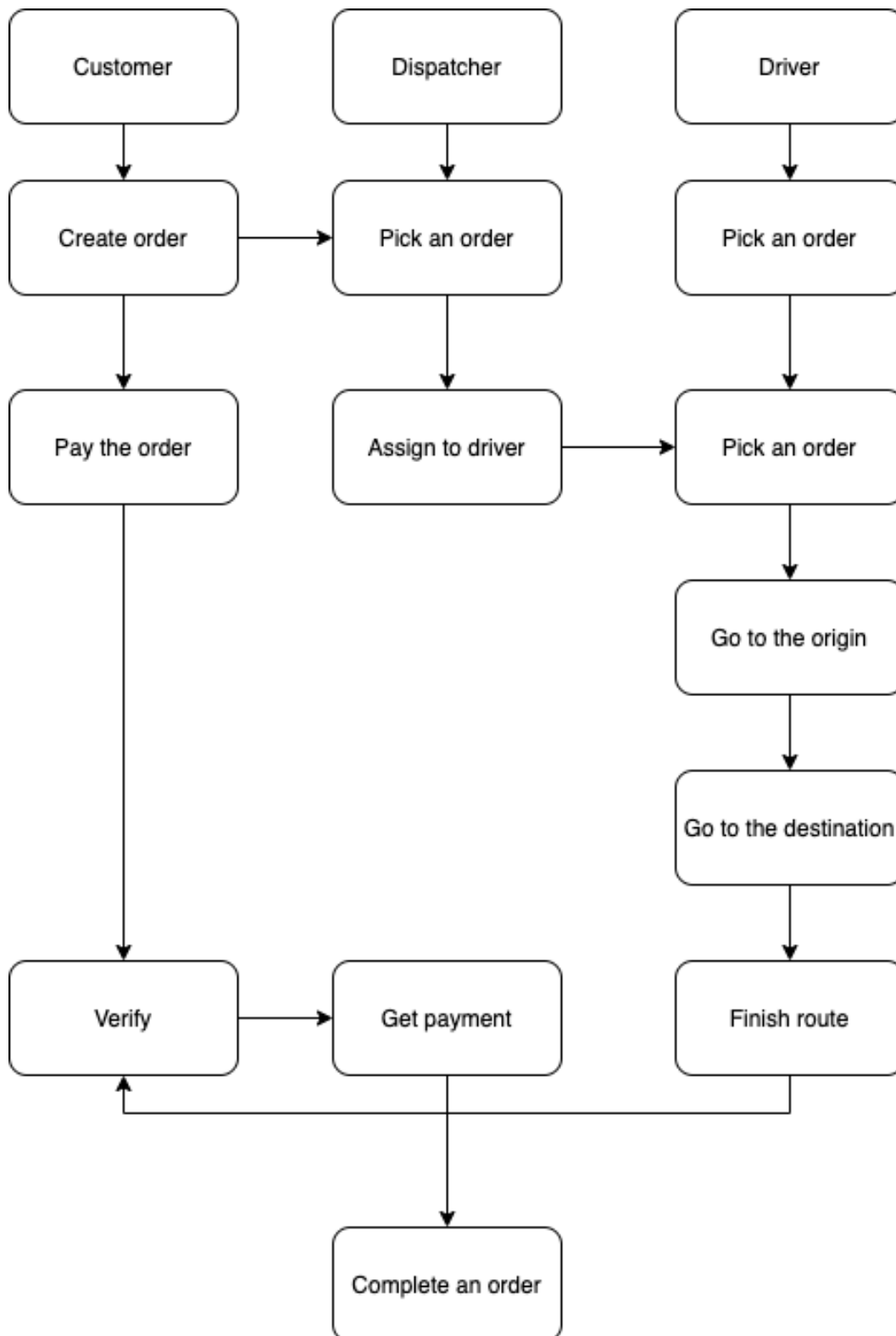


FIGURE 3.4: Order lifecycle

3.5 Libraries and frameworks

3.5.1 Google Maps

With the Maps SDK for iOS, you can add maps based on Google maps data to your application. The SDK automatically handles access to the Google Maps servers, map display, and response to user gestures such as clicks and drags. You can also add markers, polylines, ground overlays and info windows to your map. These objects provide additional information for map locations, and allow user interaction with the map.

Maps play a significant role in our application, and the functionality of Apple's built-in maps was not enough. First of all, we use the Google framework when creating an order to select the items of loading and unloading accurately. There is an autocomplete functionality when entering the address, which is very convenient for the user. We also use maps on the detailed order overview page to show the entire route and the current location of the cargo on the map. In the future, I would like to implement my own navigation in the application, but at the moment, it is enough to simply redirect the user to a third-party application with a calculated route. This can also be done through the Google framework.

3.5.2 Stripe

The Stripe iOS SDK makes it quick and easy to build an excellent payment experience for the iOS app. They provide powerful and customizable UI screens and elements that can be used out-of-the-box to collect users' payment details. They also expose the low-level APIs that power those UIs so that you can build fully custom experiences.

3.5.3 Twilio

Twilio Messaging is an API to send and receive SMS, MMS, OTT messages globally. It uses intelligent sending features to ensure messages reliably reach end-users wherever they are. Twilio has SMS-enabled phone numbers available in more than 180 countries.

We use it on our backend side to provide a verification code for the user for login or registration.

3.6 Backend

3.6.1 Overview

At the project planning stage, the question arose as to which platform to choose for the backend. The idea was to use Firebase, as it is one of the simplest options, with enough functionality for authentication through various social networks and the ability to store objects in a database. I learned that it is common practice for startups to use Firebase to get started. However, as soon as the application begins to grow, using Firebase will be very expensive. Sooner or later, I will have to abandon it and develop my own server, so I decided not to do unnecessary work and create a server on Dnango at once.

3.6.2 Django

Since I already had some experience with django while studying, I decided to use it. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

3.6.3 Rest Framework

REST is a loosely defined protocol for listing, creating, changing, and deleting data on your server over HTTP. The Django REST framework (DRF) is a toolkit built on top of the Django web framework that reduces the amount of code you need to write to create REST interfaces.

In my project, I use DRF to synchronize mobile applications with the server and database to retrieve, save or update some information about the state of the application. To do this, I implemented the following queries.

Since the application is divided into three different flows, I created a custom user class that contains information about the user type. The type can be Customer, Driver or Manager

For the basic functioning of the application, I have developed the following queries.

All user types

- **/order/action/list**
This query returns a list of orders associated with that user.
- **/order/action/update**
Used to update any information about order. For example: assign carrier, assign driver, change status.

Managers only

- **/truck/action/add**
It takes information about the truck, such as make, model number, and number. Assign this object to the current manager's ID and save it to the database.
- **/truck/action/list**
Give a list of trucks related to current manager
- **/driver/action/getInvitationCode**
This request is used to connect the driver to its manager. It returns a unique invitation code that the manager must provide to drivers to connect them to his company
- **/driver/action/list**
Returns the list of user id of drivers connected to current manager.

Drivers only

- **/driver/action/connect**
Takes invitation code as input and returns success in case invitation code is valid

Customers only

- **/order/action/create**
Takes all the necessary data from the user and creates an order and adds it to the database.

3.6.4 Authentication

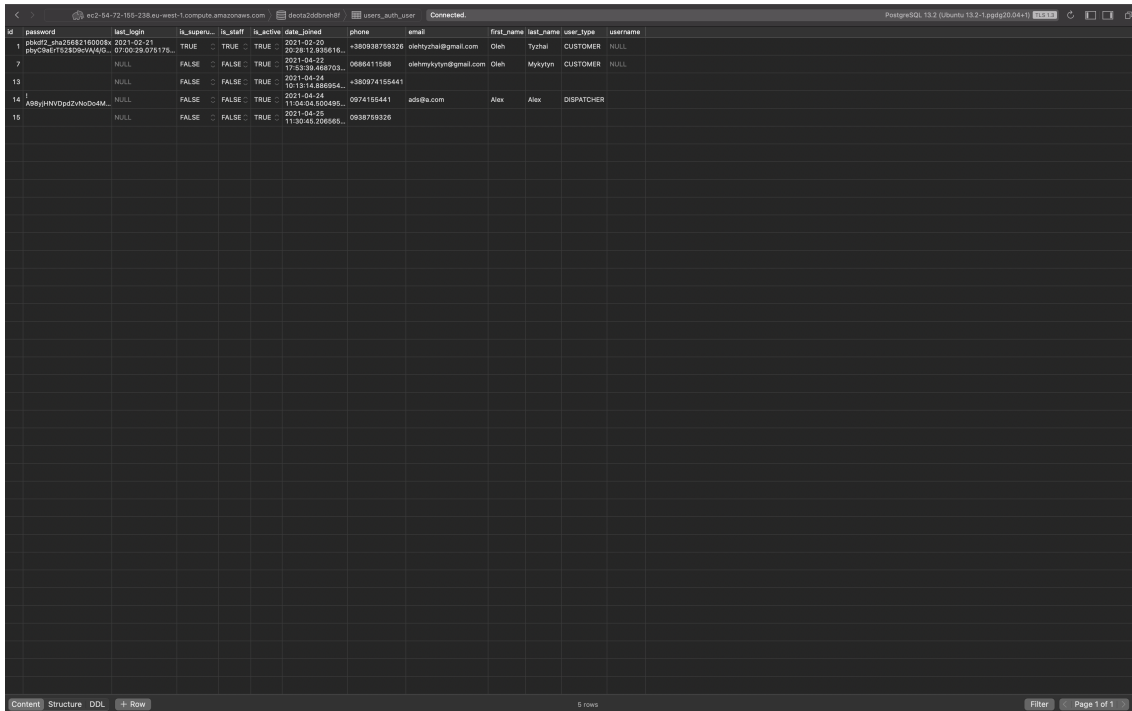
I decided to use the SMS confirmation method for authorization in the application. Three requests were created in the application for authorization:

- **/auth/action/generateOTP**
Takes the user's phone number as input randomly generates a verification code, and sends it to the phone number.
- **/auth/action/verifyOTP**
This request takes the phone number and confirmation code as input, and if the code is valid, the request will respond to tokens (access and updates) and the user object. There is no separation between login and registration in the application, so after receiving the response, we check whether the user type is already assigned to this user.
- **/auth/action/refreshToken**
We use JWT for authorization. The JSON Web Token is an open standard (RFC 7519) that defines a compact and autonomous way to transfer information between parties as a JSON object securely. This information can be verified and trusted because it has a digital signature. The JWT can be signed using a secret (with the HMAC algorithm) or a public / private key pair using RSA or ECDSA.

JWTs have an expiration date, so in such cases, we update the token to regain access to it without logging in again. And this request accepts the update token and returns an access token in response.

3.6.5 Heroku

Heroku is a container-based cloud platform used to deploy, manage, and scale modern applications. I needed a venue to host a Django server on the web, and in my opinion, Heroku is the most affordable option for startups. There is an opportunity to choose a free plan with only minor restrictions.



id	password	last_login	is_superuser	is_staff	is_active	date_joined	phone	email	first_name	last_name	user_type	username
6	pbYCR6FTS3SD9cVAAJG...	2021-02-21 07:00:29.078176...	TRUE	TRUE	TRUE	2021-02-20 20:28:12.838616...	+380928709920	olehyzhai@gmail.com	Oleh	Tyzhai	CUSTOMER	NULL
7	NULL	NULL	FALSE	FALSE	TRUE	2021-04-23 17:53:39.468703...	0686411588	olehmykyry@gmail.com	Oleh	Mykyryn	CUSTOMER	NULL
13	NULL	NULL	FALSE	FALSE	TRUE	2021-04-24 10:13:14.888954...	+380974155441	NULL	NULL	NULL	NULL	NULL
14	asBjyHNVdpsZNoD04M...	NULL	FALSE	FALSE	TRUE	2021-04-24 11:04:54.500490...	0974155441	adi@a.com	Alex	Alex	DISPATCHER	NULL
15	NULL	NULL	FALSE	FALSE	TRUE	2021-04-25 11:30:45.266965...	0938759325	NULL	NULL	NULL	NULL	NULL

FIGURE 3.5: Postico app

3.6.6 PostgreSQL

PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

In fact, I did not think long about which database to use and do not see much difference between them because Django has a wrapper that works with any database, so I chose postgres only because it has a cool application to view the tables of database - Postico.

Chapter 4

Future plans

4.0.1 Documentation

Each shipment entails a large number of documents. And to simplify the process for everyone, I would like to implement electronic document management. If we collect all the necessary information from customers and contractors, we could generate this document in the application and at each stage of transportation to receive a signature from a person or in the application or through a digital signature. Many truckers are afraid or do not trust to pass their driver's license and truck documents to accountants to fill out papers, and because of this, stressful situations often arise on both sides. I am sure that electronic document management would solve this problem completely.

4.0.2 Contacts feature

Good communication plays a significant role not only in transportation but also in any field. In order for the customer, carrier, and driver to be able to communicate freely, I want to develop chat functionality in the app. In my opinion, it is better to separate communication of the customer with the dispatcher and with the driver as the customer can discuss with the driver features of a place of loading, which is not interesting for the dispatcher and will only spam him. Also, for greater security, chats appear only when the order has been picked by manager.

4.0.3 Carrier statistics

Any company needs to see and understand what profit they will receive. In the appendix, I would like to develop a separate statistics page for the company with general information and separately for each driver or truck, so that the employer understands which of the drivers brought a large profit to the company and which a small one.

4.0.4 Truck integration

To improve the statistics and better control over the drivers, I want to develop a device that connects to the truck's CAN tire and receives information about the fuel consumption of the car, speed, and probably other information. It will help determine as accurately as possible all the costs and net income of the company for a particular order.

Chapter 5

Summary

At the initial stage of the thesis, I set myself the goal to develop a working prototype of a transport process management system. This was quite difficult, as there can be a huge number of functions, and the development of such systems can take years with a large team. To finish the basic version of the application I decided to cut some functionality which in my opinion is less important. I did not develop a mechanism for tracking orders, did not make a mechanism for secure payment and many other little things. However, I believe that I have already taken a significant step for the releasing my application.

Bibliography

- [1] What is Django
<https://www.djangoproject.com/>
- [2] Django rest framework
<https://realpython.com/lessons/building-drf-overview/>
- [3] Json Web Token
<https://jwt.io/introduction>
- [4] PostgreSQL
<https://www.postgresql.org/>
- [5] Heroku official website
<https://www.heroku.com/about>
- [6] Uber Freight users feedback
<https://www.truckingoffice.com/blog/what-do-you-think-about-uber-freight/>
- [7] Google Maps SDK overview
<https://developers.google.com/maps/documentation/ios-sdk/overview>
- [8] Stripe
<https://github.com/stripe/stripe-ios>
- [9] Twilio
<https://www.twilio.com/the-current/what-is-twilio-how-does-it-work>
- [10] Freight Shiping
<https://www.freightquote.com/define/what-is-freight-shipping/>