  
**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ФАКУЛТЕТ КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ**

**Дипломна работа на тема**:

**СИСТЕМА ЗА СЪЗДАВАНЕ И ОБРАБОТКА НА ПОРЪЧКИ В РЕСТОРАНТ**

|  |  |
| --- | --- |
| **София 2025** | **Дипломант:**  **Калоян Каменов Борисов КСИ, 121324008**  **Дипломен ръководител:**  **Ас. Иво Гергов** |

Съдържание

[Списък с фигури 5](#_Toc199882688)

[I. Постановка на дипломната работа, цели и задачи. 6](#_Toc199882689)

[1.1 Анализ на текущото състояние 6](#_Toc199882690)

[1.2 Цели и задачи 6](#_Toc199882691)

[1.3 Ограничения 7](#_Toc199882692)

[1.4 Използвани технологии 7](#_Toc199882693)

[1.4.1 Python 7](#_Toc199882694)

[1.4.2 Django 8](#_Toc199882695)

[1.4.3 Django REST Framework 8](#_Toc199882696)

[1.4.4 Kotlin 8](#_Toc199882697)

[1.4.5 Android 8](#_Toc199882698)

[1.4.6 Bluetooth 9](#_Toc199882699)

[1.4.7 ESP32 9](#_Toc199882700)

[1.4.8 Docker 9](#_Toc199882701)

[III. Функционално описание 10](#_Toc199882702)

[IV. Програмна реализация. 11](#_Toc199882703)

[V. Ръководство за използване и примери за употреба.  12](#_Toc199882704)

[VI. Заключение 13](#_Toc199882705)

[Съкращения 14](#_Toc199882706)

[Източници 15](#_Toc199882707)

# Списък с фигури

Не са намерени елементи за списък на фигурите.

# Постановка на дипломната работа, цели и задачи.

## 1.1 Анализ на текущото състояние

Ресторантьорството е един добре познат бизнес, който има представителства във всеки град. Всеки човек е бил в поне един ресторант, където най-вероятно е открил най-големите недостатъци на бизнеса.

Ресторантите са добре установена формула. Имат кухня, където се приготвя храната, имат сервитьори, които обикалят из салона, вземат поръчките на клиентите и носят поръчаното. Най-голямото забавяне между влизането в ресторанта и започването на приготвянето на поръчката е времето, което отнема на сервитьора да стигне до масата и да вземе поръчката Ви. Това време, може да бъде много, а може да бъде и малко. Никога не можем да бъдем сигурни, тъй като това зависи от броя сервитьори в ресторанта, натовареността им, а и от човешката грешка. Много често се случва, особено при висока натовареност, сервитьор да не посрещне нов клиент, заради това, че е зает с обслужването на съществуващ клиент.

Тук идват автоматизираните системи за поръчка. Такива системи не са непознати. Напоследък започват да се забелязват QR кодове, които позволяват на клиента да зареди менюто на ресторанта на своето мобилно устройство. По този начин, клиентът може предварително да реши какво иска да поръча. Тази система ускорява времето за поръчка, но отново имаме чакането докато се освободи сервитьор, който да ни обслужи.

Съществуват друг вид такива системи, които добавят възможност за поръчка, след сканиране на QR кода. При тези системи, се прави поръчка чрез онлайн формуляр, в който автоматично се попълва номерът на масата, след като е сканиран QR кодът. Тези системи позволяват директна поръчка към кухнята. При тях, храната ни започва да се приготвя в момента, в който я изберем. Големият недостатък при този вид системи, е че зловредни потребители могат да създават фалшиви поръчки от собствения си дом. Достатъчно е потребителят да е посетил ресторанта веднъж, и вече те могат да използват същия QR код и да правят поръчки за дадена маса, без те да бъдат там.

Тази разработка ще предложи надграждане на автоматичните системи за приемане на поръчки, като основната цел ще бъде премахване на поръчките, направени от потребители, които не се намират в ресторанта. Ще бъдат използвани технологии за сканиране на маяци, които ще потвърждават дали потребителят се намира в ресторанта или някъде извън него.

## 1.2 Цели и задачи

За тази разработка са поставени няколко цели:

* Да се реши проблемът с фалшивите поръчки при автоматичните системи за поръчка
* Да се предостави удобно приложение за потребителите
* Да се показват създадените поръчки по удобен за четене начин

За да бъдат постигнати тези цели, трябва да бъде изградена система, която да бъде лесна за достъпване от потребителите и да предоставя удобен начин за създаване на поръчка към кухнята на даден ресторант. Системата трябва да бъде достъпна от мобилното устройство на потребителя, тъй като всеки потребител използва собствените си устройства с най-голяма лекота.

Освен мобилното приложение, към системата трябва да бъде изграден сървър, който да приема поръчките и да ги предоставя на кухнята по удобен за персонала начин. Също, трябва да бъдат създадени маяци, които да бъдат разположени по масите в ресторанта, така че когато потребител заеме дадена маса, поръчката да бъде създавана директно за конкретната маса, без необходимост от въвеждане на допълнителни данни от страна на потребителя.

Задачите, които можем да поставим за тази разработка са:

* Да се създаде мобилно приложение, базирано на Kotlin
* Мобилното приложение да използва BLE маяци, чрез които да определя масата на потребителя
* Да се създаде сървърно приложение, което да приема поръчките от клиентите
* Да се създаде интерфейс, който да показва на готвачите и сервитьорите поръчките и техните статуси

## 1.3 Ограничения

При разработката на дадената система трябва да се съобразим с ред ограничения, които не могат да бъдат избегнати. Тези ограничения произлизат от необходимостта за защита на личните данни на потребителите. Това означава, че мобилното приложение трябва да се съобрази с желанията на потребителите, като през това време не предоставя техните данни и също не злоупотребява с тях.

При разработката на мобилното приложение, трябва да се положи особено внимание върху сигурната комуникация между приложението и сървъра. При незащитена комуникация е възможно личните данни на потребителя да бъдат засечени от недоброжелатели и използвани по неоторизиран начин.

## 1.4 Използвани технологии

### 1.4.1 Python

Сървърът, който ще обслужва системата ще бъде написан на Python. Този език предоставя начин за бързо прототипизиране на система, благодарение на големия набор от библиотеки, които поддържа, както и на голямото количество програмисти, които го използват и са активни във форуми за взаимопомощ.

### 1.4.2 Django

Django е библиотека за изграждане на сървъри, които да обслужват данни. Предоставя голям набор от инструменти, които могат да бъдат използвани за лесно създаване на сървър. Основната цел на библиотеката е да предостави среда, която да премахне голямата част от често срещаните проблеми при разработка на сървърни приложения и да улесни и ускори работата на програмистите [1].

Библиотеката е с отворен код, като програмистите са окуражавани да се включват в нейната разработка, като целта е да бъде направена колкото се може по-достъпна. Тя предоставя лесен начин за начинаещи програмисти да пишат уеб приложения, а опитните програмисти могат да се възползват от широкият набор от функционалности, които библиотеката предлага.

### 1.4.3 Django REST Framework

Django REST Framework или DRF представлява библиотека, която разширява способностите на Django, като добавя възможност за създаване на RESTful API. Библиотеката предоставя удобен начин за създаване на API, което да поддържа всички заявки, като GET, POST, PUT, DELETE и др. Библиотеката също предоставя удобен начин за сериализация на обекти, които да могат да бъдат изпращани между клиента и сървъра [2].

### 1.4.4 Kotlin

Kotlin представлява език за програмиране, който може да бъде използван за създаване на приложения за разнообразни платформи. Езикът е направен да работи, използвайки Java и така постига тази платформена независимост. Стандартната библиотека за Kotlin стои директно върху Java библиотеки.

Езикът е широко разпространен при разработка на мобилни приложения, тъй като надгражда Java, която е била най-разпространена, преди Kotlin да се появи. Kotlin предоставя ясен синтаксис и надгражда Java с разширена сигурност [3].

### 1.4.5 Android

Android е най-разпространената операционна система за мобилни устройства. Разработва се от Google, като е базирана върху Linux. В основата си Android е операционна система с отворен код, разработвана от Open Handset Alliance, но популярнта версия на операционната система, която се изпълнява върху почти всички устройства е тази, разработена от Google, която обаче не е с отворен код [4].

### 1.4.6 Bluetooth

Bluetooth представлява технология, която се използва за трансфер на данни между устройства на кратки разстояния. Технологията се е установила като стандарт при комуникацията между устройства като мобилен телефон и слушалки или компютри и мишки или клавиатури. Технологията е доста стара, но нейното развитие не спира. В днешно време, устройствата, които използват Bluetooth, вече могат да комуникират с доста висока скорост и да не използват прекалено много от батерията си. Разбира се, при приложения, които изискват бързо предаване на големи обеми данни безжично, Wi-Fi остава предпочитаният избор [5].

Съществува вариант на Bluetooth, който използва минимално количество енергия, наречен Bluetooth Low Energy или BLE. Тази технология работи много подобно на нормалния Bluetooth, с основната разлика, че при нея устройствата са в спящ режим през повечето време. Те се събуждат само ако някой се опита да инициира връзка с тях [6]. Това ги прави много полезни като маяци, които да обявяват съществуването си и близостта си до устройството, което се опитва да се свърже с тях.

### 1.4.7 ESP32

ESP32 са серия от микроконтролери, разработени от Espressif, които придобиват популярност благодарение на своята ниска цена и висока енергиина ефективност. Тези микроконтролери са използвани за най-различни любителски проекти, тъй като предоставят голямо количество методи за безжична комуникация, която позволява да направим умни устройства вкъщи [7]. Заради тази популярност, тези микроконтролери са добре документирани и много от случаите на употреба, които можем да имаме, са вече разработени и достъпни за използване.

### 1.4.8 Docker

Docker представлява платформа, използвана за разработка и изпълнение на приложения. Платформата ни предоставя начин да разкачим приложението от инфраструктурата, така че да можем да предоставим софтуер бързо, без много конфигуриране. Docker предоставя контейнери, които представляват олекотени среди, в които да се изпълнява приложението ни. Тези среди могат да бъдат споделяни лесно, което улеснява процеса на работа [8].

# III. Функционално описание

# IV. Програмна реализация.

# V. Ръководство за използване и примери за употреба.

# VI. Заключение

# Съкращения

8

API - Application Programming Interface 8

BLE - Bluetooth Low Energy 9

DRF - Django REST Framework 8

QR - Quick Response 6

REST - Representational State Transfer 8

# Източници

|  |  |
| --- | --- |
| [1] | Django, „Meet Django,“ [Онлайн]. Available: https://www.djangoproject.com/. |
| [2] | A. H. Maki, „Create REST API using Django REST Framework,“ [Онлайн]. Available: https://medium.com/@ahmalopers703/getting-started-with-django-rest-api-for-beginners-9c121a2ce0d3. |
| [3] | J. Dwivedi, „Mastering Kotlin for Android: A Deep Dive Learning Guide,“ [Онлайн]. Available: https://medium.com/@jaidwivedi20/mastering-kotlin-for-android-a-deep-dive-learning-guide-5db852c277e6. |
| [4] | E. Mixon, „Android OS,“ [Онлайн]. Available: https://www.techtarget.com/searchmobilecomputing/definition/Android-OS. |
| [5] | B. Collins, „What is Bluetooth,“ [Онлайн]. Available: https://www.forbes.com/sites/technology/article/what-is-bluetooth/. |
| [6] | B. Proctor, „Bluetooth vs Bluetooth Low Energy,“ [Онлайн]. Available: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy. |
| [7] | Espressif, „ESP32 SoC,“ [Онлайн]. Available: https://www.espressif.com/en/products/socs/esp32. |
| [8] | Docker, „Docker overview,“ [Онлайн]. Available: https://docs.docker.com/get-started/docker-overview/. |