НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

***з дисципліни "Основи веб-програмування"***

Виконав: Симонюк Володимир Павлович

Група: КП-83

Допущено до захисту

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 семестр 2019/2020

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

|  |  |
| --- | --- |
| Узгоджено  Керівник роботи    \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ | ЗАХИЩЕНА "\_\_"\_\_\_\_\_\_\_\_\_2019р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ |

**Веб-додаток онлайн ресторан**

Виконавець роботи

Симонюк Володимир Павлович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019р.

Зміст

[**1. Аналіз предметної галузі**](#_b73pd3816f8p) **4**

[1.1. Аналіз вимог до функціональності програмних засобів](#_qz6vreit6wm5) 4

[**2. Аналіз мов програмування та технологій розроблення**](#_yisyng926cdy) **6**

[2.1. Мова програмування JavaScript](#_fh0j2unhluf2) 6

[2.2. Фреймворк NodeJS](#_ayd32pcv0jlc) 6

[2.3. Фреймворк ExpressJS](#_btrfiigetrfo) 6

[2.4. База даних MongoDB](#_hul0g052l08h) 7

[2.5. Файловий хостинг Cloudinary](#_pwcl1y9n0k6t) 7

[2.6. Веб-хостинг Heroku](#_tlcp7442cayj) 7

[2.7.](#_19uzz4vvgf99) Фреймворк Angular 8 7

2.8. Бібліотека node-telegram-bot-api 8

2.9. Сервіс для розсилки повідомлень SendGrid 8

2.10. Набір стилів Materialize 8

[**3. Опис розроблених програмних засобів**](#_1pur3ko1vvgc) **9**

[3.1. Загальна структура системи](#_kisqmwt7gkz) 9

[3.2. Структура бази дани](#_ny4p6qq0v8a9)х 11

[**4. Аналіз розроблених програмних засобів**](#_g0l8gqqu407h) **13**

[4.1. Особливості реалізації](#_5ajysoaxe2s5) 13

[4.2. Дизайн та вміст вікон](#_c8eyxdz8sp3u) 14

[**Висновки**](#_6dbqg2jsslr5) **21**

# 1. Аналіз предметної галузі

## 1.1. Аналіз вимог до функціональності програмних засобів

В процесі аналізу вимог до системи було виділено такі функціональні вимоги

1. Наявний веб-інтерфейс користувача. Графічні елементи використані відповідно до їхнього основного призначення.
   1. Наявно декілька сторінок або станів односторінкового сайту і веб-форм.
   2. Дотримується єдиний стиль для всіх веб-сторінок. Всі елементи стилізовано. Кнопки створення, редагування та видалення виділені спеціальними стилями (кольорами чи іншим способом).
   3. На сайті наявні гіперпосилання між сторінками\розділами\сутностями.
   4. Наявна навігація по сайту у вигляді меню чи навігаційної панелі.
2. Можливість створення, редагування та видалення даних.
   1. На формі створення обов’язкові поля мають значення за замовчуванням.
   2. Наявна форма редагування даних має мати вже заповнені поля із попередніми значеннями. Є можливість відмінити оновлення даних.
   3. Можна видаляти дані (з підтвердженням цієї дії).
3. Використання графічних списків та таблиць для відображення колекцій даних.
4. Можливість пошуку сутностей.
   1. Сайт дозволяє шукати сутності по одному чи декількох полях.
   2. Пошук повинен відбуватися по підрядку з ігноруванням регістру символів.
   3. Показувати по яких полях і що шукається, загальну кількість знайдених результатів.
   4. При невдалому пошуку виводити повідомлення про відсутність шуканих елементів.
5. Програма взаємодіє з базою даних. Запити до бази винесено у спеціальний окремий модуль (вид сховища).
   1. База даних містить мінімум 4 таблиці (колекції).   
      Реляційні таблиці в БД приведені до 1-ї та 2-ї нормальних форм.
   2. Між таблицями (колекціями) бази даних є one-to-many
6. Програма має можливість завантаження файлів з файлової системи користувача, їх збереження у сховище даних.
   1. Завантажені користувачем зображення можна переглядати на сторінках сайту.
   2. Файли зберігаються у базі даних або на спеціальному віддаленому файловому хостингу.
7. Наявна реєстрація користувачів у системі. Паролі користувачів не зберігаються у відкритому виді, а хешуються обраним алгоритмом хешування рядків.
   1. Наявна форма аутентифікації. Пароль користувача при введені у форму прихований (використано спеціальний елемент). Сторінка цієї форми має сповіщення про некоректність введених користувачем даних.
   2. Програма авторизує доступу до даних по користувачах. У базі є дані, що належать конкретному користувачу (напр, авторство сутностей).
   3. Є мінімум 2 ролі користувачів. Обов’язкова роль адміністратора, що може керувати іншими користувачами.
   4. Користувач може вийти з сайту (logout). Наявна персональна сторінка користувача.
8. У сервера є можливість доступу до даних через API.
   1. Частина API шляхів (endpoints) надають доступ лише після авторизації.
   2. На сайті є веб-сторінка із коротким описом API.
9. Всі вхідні дані від користувача та з файлів перевіряються, помилки обробляються, користувачу відображається інформація про помилки.
10. Використання нотифікацій за допомогою ботів.
    1. Можна підключити телеграм бот для отримання персоналізованих даних від бота та автоматичних нотифікацій підписаних користувачів при певних діях на сайті.
11. SPA сайт із використанням Front-end фреймворка Angular 8.
12. Використання JWT для SPA та Social Login за допомогою OAuth2

# 2. Аналіз мов програмування та технологій розроблення

## 2.1. Мова програмування JavaScript

Мова була використана для створення сценаріїв веб-сторінок, надала можливість на стороні сервера взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з базою даних, змінювати структуру та зовнішній вигляд веб-сторінки.

## 2.2. Фреймворк NodeJS

Node.js - програмна платформа, що перетворює JavaScript з вузькоспеціалізованої мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення, підключати інші зовнішні бібліотеки, написані на різних мовах.

У нашому випадку веб-сервер та телеграм-бот були повністю виконані на цій платформі. Найбільшим плюсом цієї платформи, як на мене, є можливість встановлювати величезну кількість найрізноманітніших бібліотек за допомогою його власного установника пакетів npm.

## 2.3. Фреймворк ExpressJS

Express.js, фреймворк web-додатків для Node.js, спроектований для створення веб-додатків і API, є стандартним каркасом для Node.js. Express у нашому проекті використовується у якості backend'у, разом з базою даних MongoDB.

## 2.4. База даних MongoDB

MongoDB - документоорієнтована СКБД з відкритим вихідним кодом, яка не потребує опису схеми таблиць, використовує JSON-подібні документи і схему бази даних. Дуже зручна для використання, підтримує ad-hoc-запити, які можуть повертати конкретні поля документів і користувацькі javacript-функції. Також гарним доповненням є те, що вона підтримує регулярні вирази.

## 2.5. Файловий хостинг Cloudinary

Cloudinary надає хмарні послуги управління медіафайлами. Це дозволяє користувачам завантажувати, зберігати, керувати, маніпулювати та доставляти зображення та відео для веб-сайтів та додатків.

## 2.6. Веб-хостинг Heroku

Heroku - хмарна платформа, що підтримує декілька мов програмування в тому числі Node.js. Веб-хостинг запускає програми клієнта у віртуальні контейнери, які виконуються у надійному середовищі виконання.

## 2.7. Фреймворк Angular 8

**Angular** — написаний на TypeScript front-end фреймворк з відкритим кодом. У нашому додатку був використаний для створення клієнтського сервера. Зручний роутинг та його зв’язка з компонентами надали змогу більш зручно розробляти додаток, також цьому посприяли велика кількість вбудованих фіч таких як RxJs, ReactiveForms, Guards (canActivate interfaces вбудовані в роутинг), HttpClient, та безліч інших чудових речей.

2.8. Бібліотека node-telegram-bot-api

Модуль Node.js для API Telegram Bot (https://core.telegram.org/bots/api). Використовується у додатку для взаємодії з telegram-bot-api

2.9. Сервіс для розсилки повідомлень SendGrid

SendGrid надає хмарну послугу, яка допомагає користувачам з доставкою електронної пошти. Служба керує різними типами електронної пошти, включаючи сповіщення про доставку, запити друзів, підтвердження реєстрації та електронні розсилки.

2.10. Набір стилів Materialize

Materialize - обгортка для Material Design - принципів дизайну сайтів від компанії Google. Ідея дизайну полягає в інтерфейсі, поведінка і вигляд якого наслідують правила поведінки і вигляду паперових карток в реальному житті. Існує також визначення, що матеріальний дизайн є візуальною чи дизайн мовою.

# 

# 

# 

# 

# 3. Опис розроблених програмних засобів

## 3.1. Загальна структура системи

Програмні засоби реалізовані у вигляді web-додатку. Структурна схема системи зображена на рис.

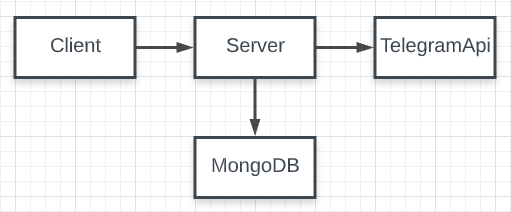
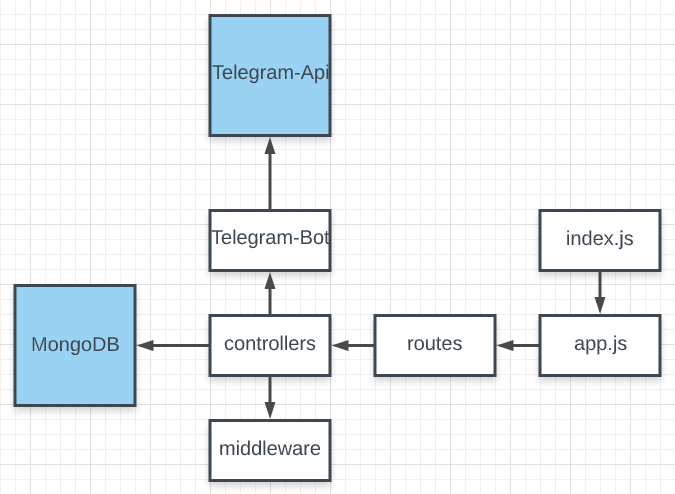


Рис. Структурна схема системи

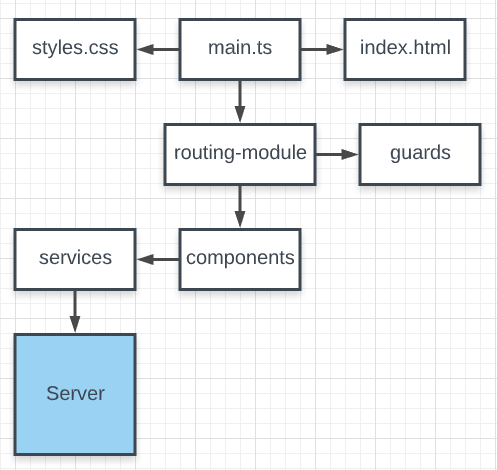
Сервер системи обробляє запити від користувачів та виконує такі функції:

1. CRUD операції для всіх моделей (колекцій)
2. Нотифікація за допомогою Телеграм-бота при зниження цін в певній категорій
3. Використання JWT для SPA та Social Login за допомогою OAuth2
4. Додаткові функції для взаємодії з БД

Діаграма залежностей модулів сервера:



Діаграма залежностей модулів Клієнта:



## 3.2. Структура бази даних

На рис. представлена ERD-діаграма структури бази даних системи.

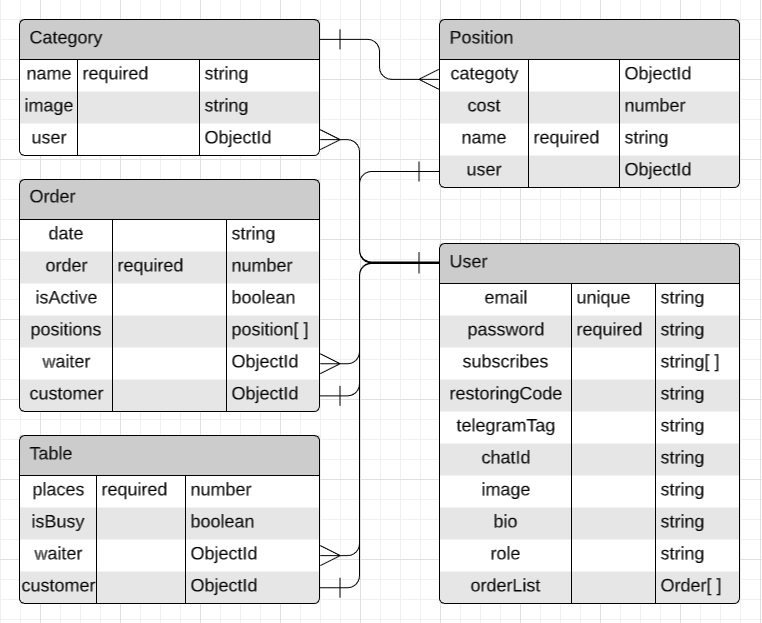


Рис. Структура бази даних

1. Таблиця користувачів

Кожна сутність представляє собою одного користувача системи який на початку роботи з додатком створює екземляр, ввіши унікальну **email** адресу та **password** до відповідної форми на стартовому вікні, **роль** користувачу надається автоматично, але адміністратор вміє змінювати її в спеціальному вікні. Також користувач містить такі поля як **TelegramTag** i **chatId** які заповнюються згодом для взаємодії з телеграм ботом який вміє сповіщати користувача про зниження ціни у категоріях які він додав в обране ( коли користувач додає категорію в обране, її масив **subscribes** отримує ІД цієї категорії ). І додаткові поля **bio** i **image** використовуються для того щоб зберігати додаткові дані про користувача, які він може додати у вкладці профілю.

1. Таблиця позицій

Екземпляр позиції являє собою одиничну страву яка не є самостійною і обов’язково відноситься до однієї із категорій ( ця залежність може бути відстежена за полем **categoryId** ).

Також позиція містить обов’язкове для заповнення поле **name**, вартість

( **cost** ). Та користувача, який неї створив, утворюючи з ним зв’язок **many-to-one.**

1. Таблиця категорій

Категорія - сутність, що групує позиції у групи ( зв’язана з ними **one-to-many** ). Містить обов’язкове для заповнення поле **name** та **зображення**, яке допоможе користувачу розпізнати категорію страв краще.

Та користувача, який неї створив, утворюючи з ним зв’язок **many-to-one.**

1. Таблиця столиків

Столик суттєво відрізняється від попередніх сутностей тим, що містить динамічні зв’язки **many-to-one** i **one-to-one,** з **офіціантом** та **гостем** відповідно. Основною інформацією у цій сутності є інформація про загальну **кількість** **місць** за столом та інформація про те **чи** **зайнятий** він наразі.

1. Таблиця замовлень

Одиничне замовлення характеризується наступними параметрами:

**дата** **створення** замовлення та його **порядковий** **номер**, які задаються автоматично на backend`і, **чи** **обслуговується** він наразі **відвідувачем**/**офіціантом**, **масив** **позицій** які до нього входять і **відвідувач** за яким це замовлення ізначально закріплений.

# 4. Аналіз розроблених програмних засобів

## 4.1. Особливості реалізації

Наш веб-додаток наповнений великою кількістю фіч та функціоналу, про основні я розповім нижче:

* авторизація відбувається з використанням PassportJS, з використанням JsonWebToken стратегії.
* всі паролі надійно хешуються з сіллю за допомогою бібліотеки bcryptJS
* користувачу надається змога відновити втрачений пароль за допомогою email’а, на який буде надіслано код підтвердження, копія цього ключа буде також зберігатися в користувачі у захешованому вигляді, і при спробі зловмисника змінити пароль користувача через АПІ, його запит буде відхилено через відсутнісь правильного коду в запиті, ввівши код правильно, у спеціальній формі користувач зможе змінити пароль
* користувачі можуть набувати одної з чотирьох ролей, і в залежності від цього отримувати різний функціонал

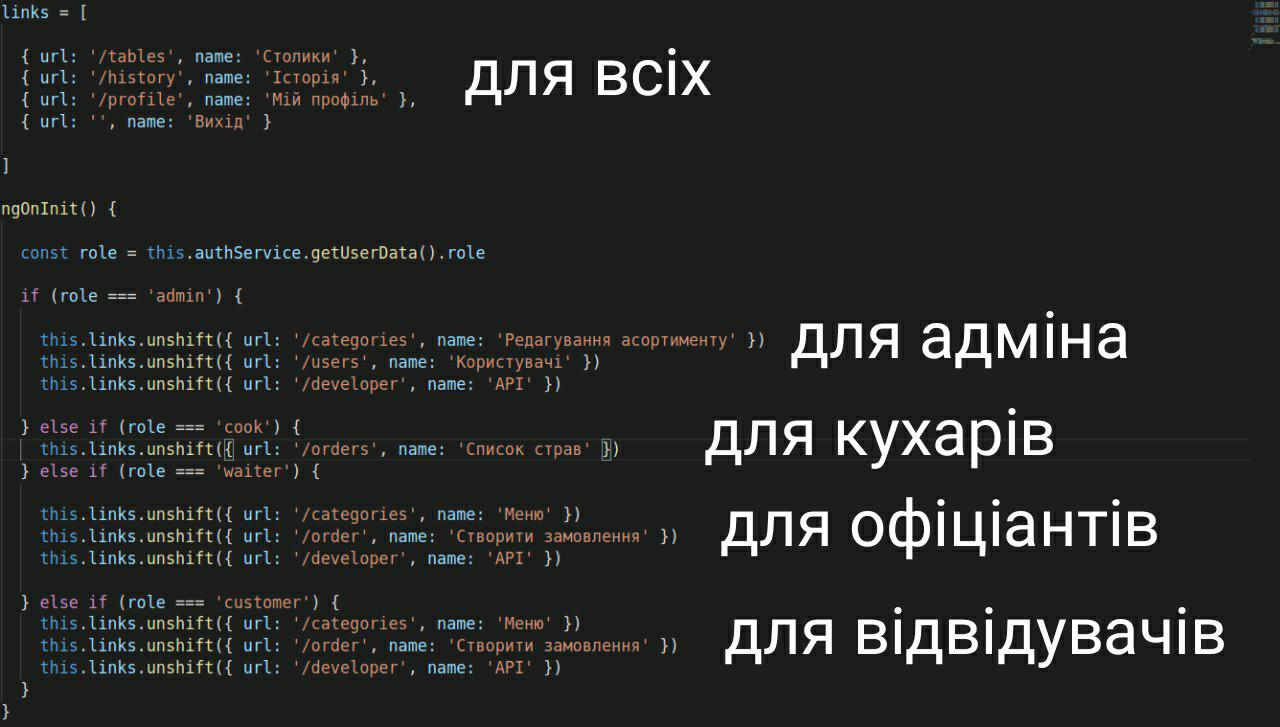


рис.4.1

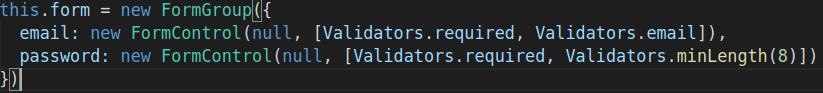
* для того щоб почати роботу з ботом потрібно на сторінці профілю ввести дійсний телеграм юзернейм, а в самому боті виконати команду ‘/synchronize’ цю команду необхідно виконати для того, щоб у базі даних, заповнити поле chatId для користувача, після цього бот зможе надсилати користувачу повідомлення
* користувач може переглядати меню через бота
* користувач може підписуватись на категорії які його цікавлять, після виконання підписки ІД категорії додається в масив підписок користувача.  
  При зменшені ціни позиції, всім користувачам, підписаним на категорію, до якої вона належить, приходять сповіщення

## 4.2. Дизайн та вміст web-сторінок

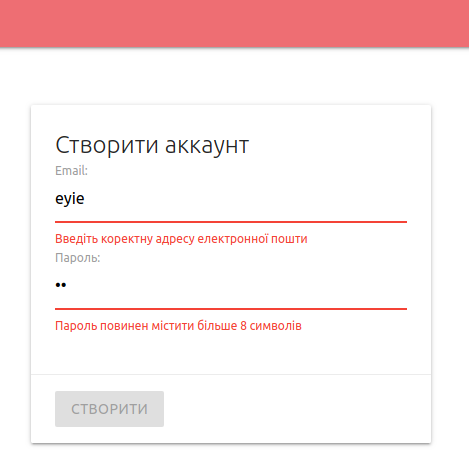
Для відображення даних система має графічний web-інтерфейс користувача, що складається з двох основних складових: реєстраційної/аутентифікаційної і загальної компонент

1.**Auth-Layout** - компонент який слугує обгоркою для двох автризаційних вікон:

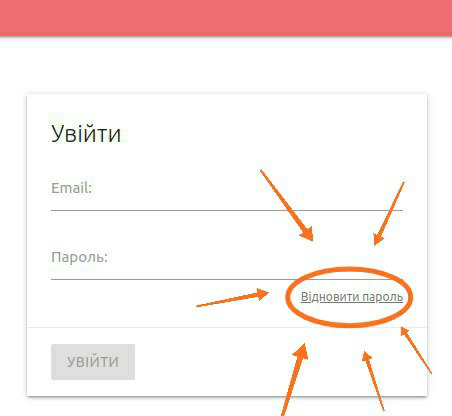
* **register-page** - сторінка, яка містить **реактивну** **форму,** заповнення якої валідними даними надає користувачу дійсний профіль, який він може використовувати для отримання доступу до всього функціоналу додатку.



Всі поля захищенні спеціальними **валідадорами,**  властивостями яких наділені поля всіх форм додатку.

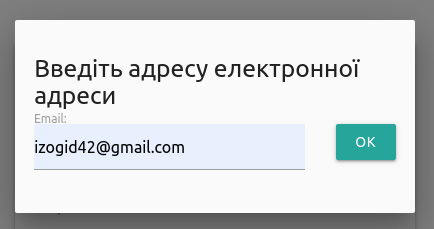


* **login-page** - сторінка виглядає аналогічно до сторінки реєстрації, відрізняються вони лише масивами валідаторів у ФормКонтролах і додатковою кнопкою, яка допомагає користувачу **відновити** **пароль**.

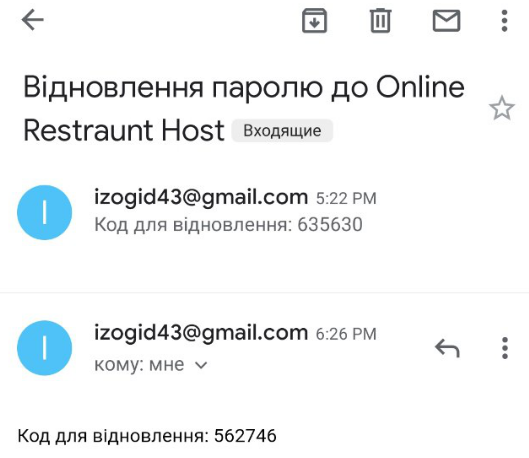


При спробі відновити пароль, вам потрібно пройти верификацію, що складається з серії модальних вікон, які приймають зазначену інформацію, після чого ввести новий пароль.

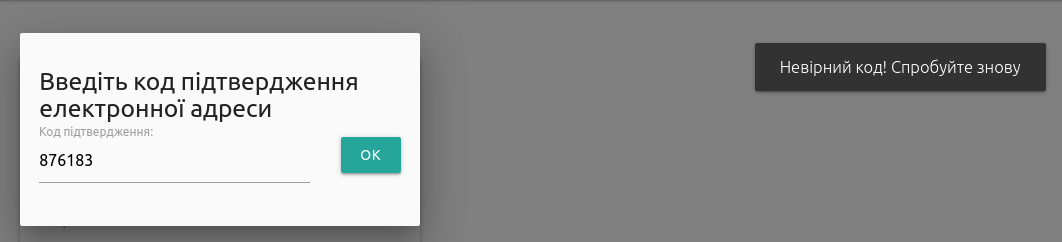
1. Ввести електроону пошту зареєстрованого користувача



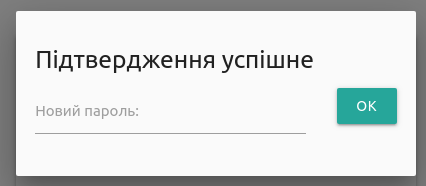
1. Ввести код, надісланий на введену адресу



Ввівши код неправильно, отримаєте наступне:



1. В зворотньому випадку надасться змога змінити пароль



2. **Site-Layout** - компонент що має навігаційну панель і футер з контактами розробника, а за допомогою плейсхолдера <router-outlet></router-outlet> реалізовується вся **маршрутизація** **додатку**, це наглядно видно на **рис.4.2**

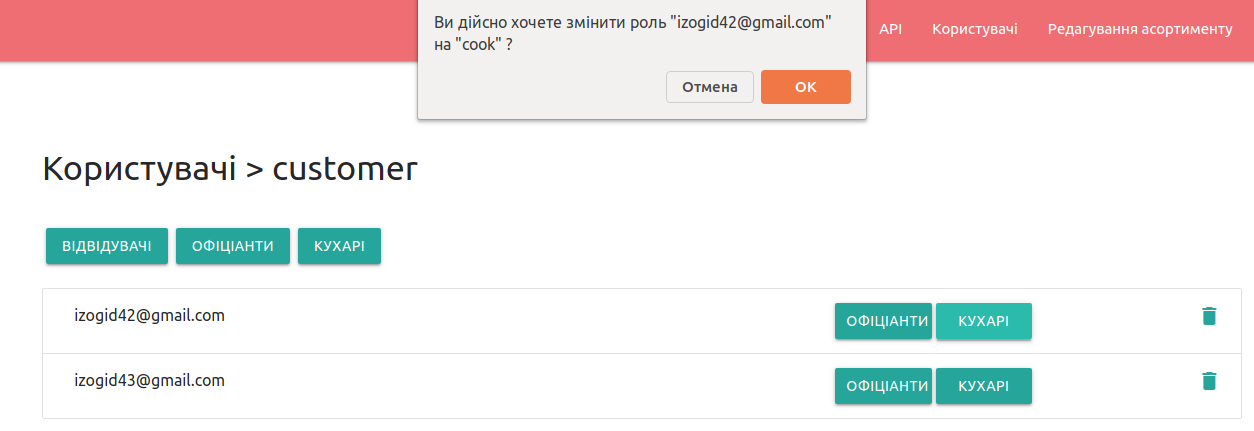
Нижче буде описаний функціонал **окремих** **компонентів**, подивитись хто має доступ до цього компоненту можна на **рис.4.1**

* **API -** веб-сторінка із коротким описом API.

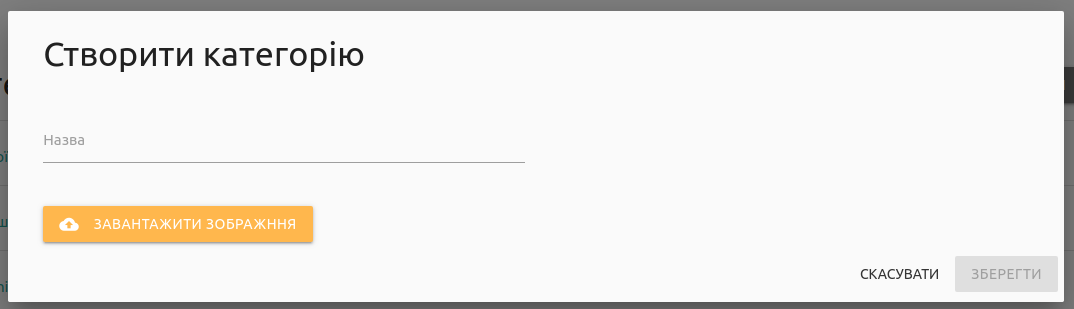


* **users-page -** компонент на якому **адміністратор** системи

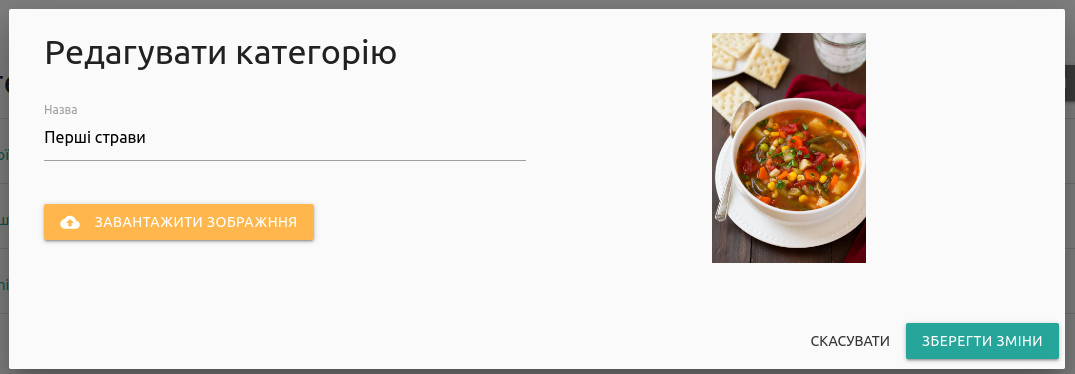
може переглядати користувачів по ролях, видаляти них, та переміщати міщ категоріями. Призначена вона для назначення керуючим, нових офіціантів та кухарів.



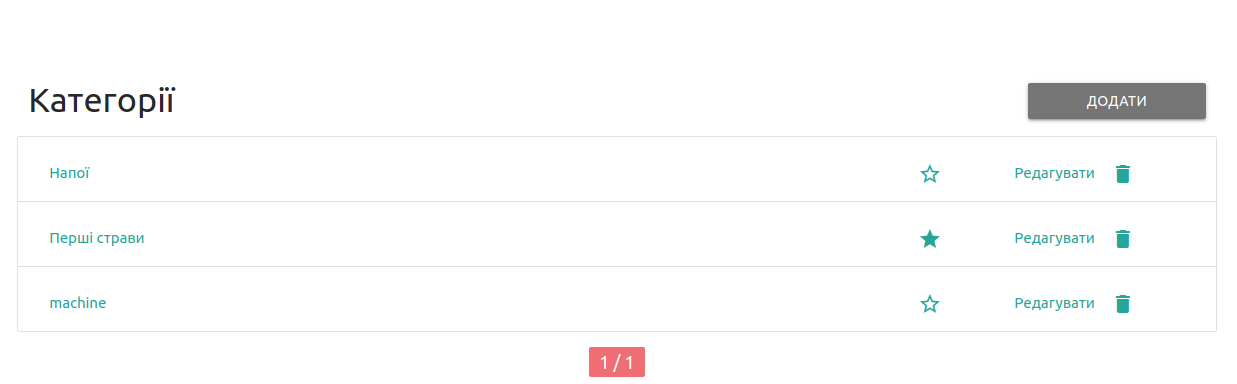
* **category-page -** сторінка з пагінацією на якій адмінстратор може **додавати**,



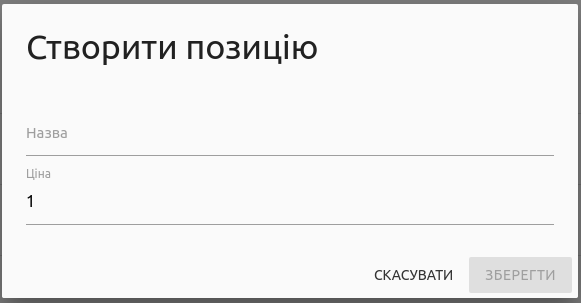
**видаляти** та **редагувати** категорії.

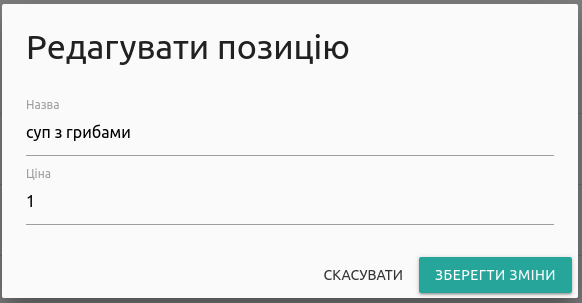


Звичайний ж користувач може лише **переглядати** категорії, та додавати їх до обраного, щоб в подальшому отримувати **push-сповіщення** через **телеграм-бота** у разі зниження ціни, на її позицію, адміністратором. Деталі про роботу бота можна прочитати в параграфі **4.1**

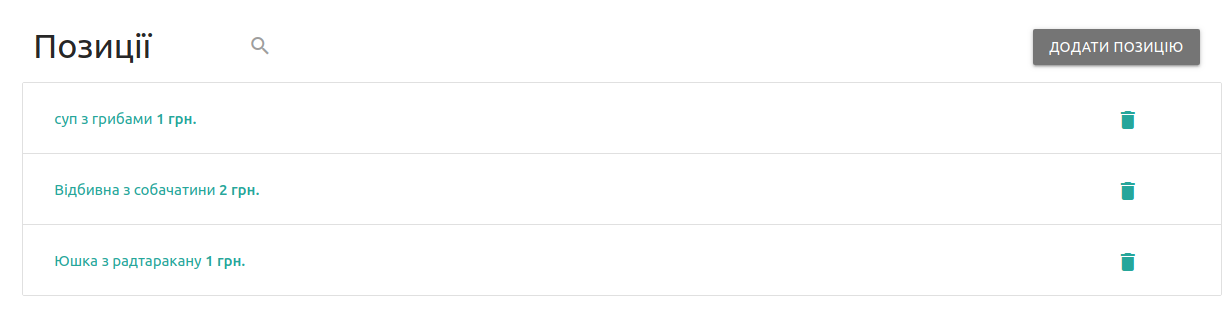
****

* **position-page -** сюди можна потрапити натиснувши на одну з категорій, сутності, відображені на сторінці, належать до батьківської категорії, операції які можуть виконувати користувачі є аналогічними до категорій

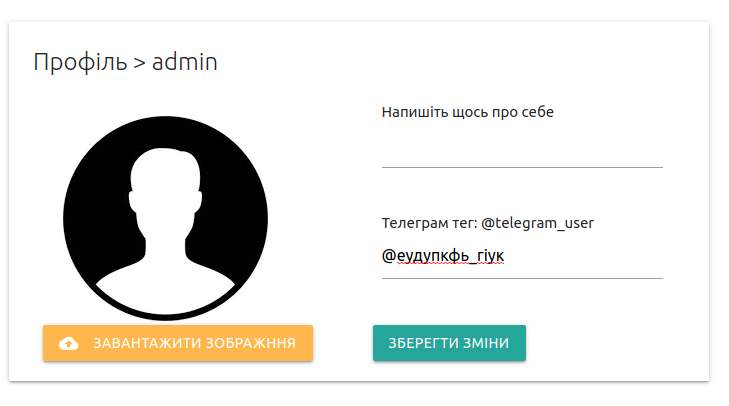




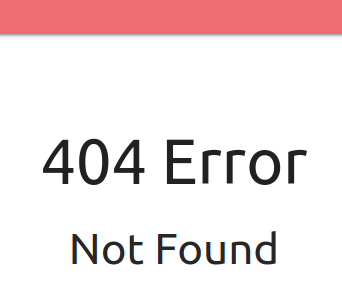
Також наявне поле пошуку, що допоможе серед всього різноманіття швидко знайти улюблені страви

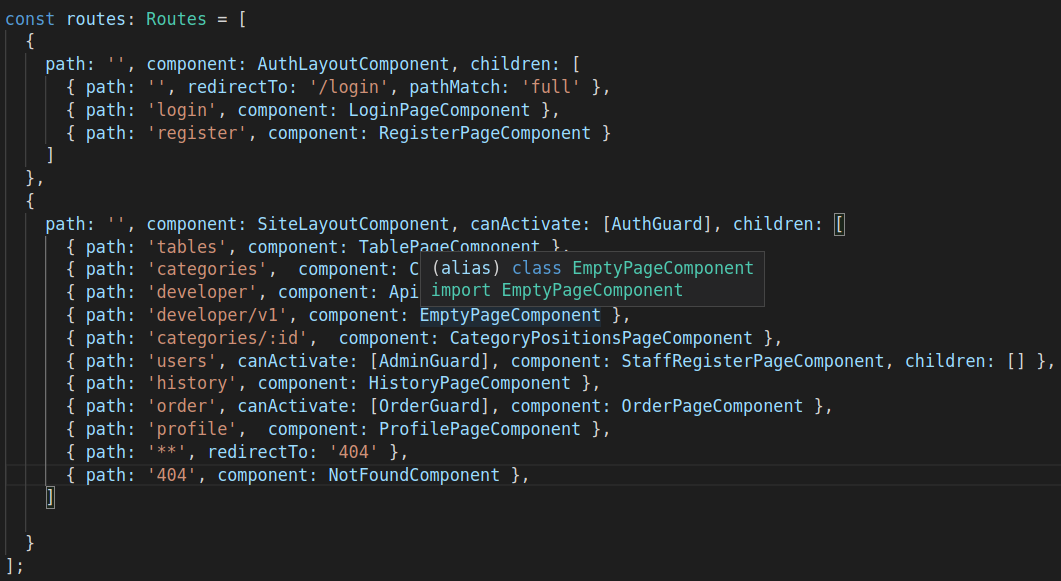


* **profile-page -** компонент на якому користувач може переглянути або відредагувати такі поля як **bio** та **telegramTag**, якраз після введення телеграм-юзернейма користувач може, за допомогою спеціальної команди, провести синхронізацію з ботом, і використовувати його функціонал.



* **not-found -** сторінка, що відображається, коли користувач неправильно вводить шлях, по якому розміщений компонент, або намагається відкрити переглянути позиції, що належать неіснуючій категорії



рис.4.2 routing-module

# Висновки

Метою даної курсової роботи було розроблення web-додатку для більш зручної взаємодії персоналу різних харчувальних закладів з їх клієнтами.

Розроблена система:

* забезпечує авторизований доступ до системи;
* дозволяє управляти категоріями страв, та одиничними компонентами меню, і столиками закладу.
* дає можливість адміністратору спростити процедуру редагування меню, зарахування персоналу, та реєстрації нових місць.

Особливу увагу під час розроблення даного програмного продукту було приділено простоті та елегантності дизайну, написанню грамотного та зрозумілого коду і, найважливіше, залученню найсучасніших технологій для реалізації додатку

Розробка виконана у повному обсязі, всі вимоги враховані, продукт протестовано. Використання розробленої системи дозволить спростити роботу як персоналу, так і відвідувачам.