

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №44

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

старший преподаватель
должность, уч. степень, звание

подпись, дата

А.В. Аксенов
инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

КУЛИНАРНЫЙ БЛОГ «Chef Blog»

по дисциплине: БАЗЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4142

подпись, дата

Т.О. Казакевич
инициалы, фамилия

Санкт-Петербург 2023

ПОСТАНОВКА ЗАДАЧИ

Целью курсовой работы является разработка веб-приложения для кулинарных рецептов, позволяющего пользователям делиться своими блюдами и просматривать уже созданные записи, а также комментировать их. Пользователям должен быть предоставлен доступ к рецептам по различным категориям, добавление в избранное, а также поиск по ключевым словам.

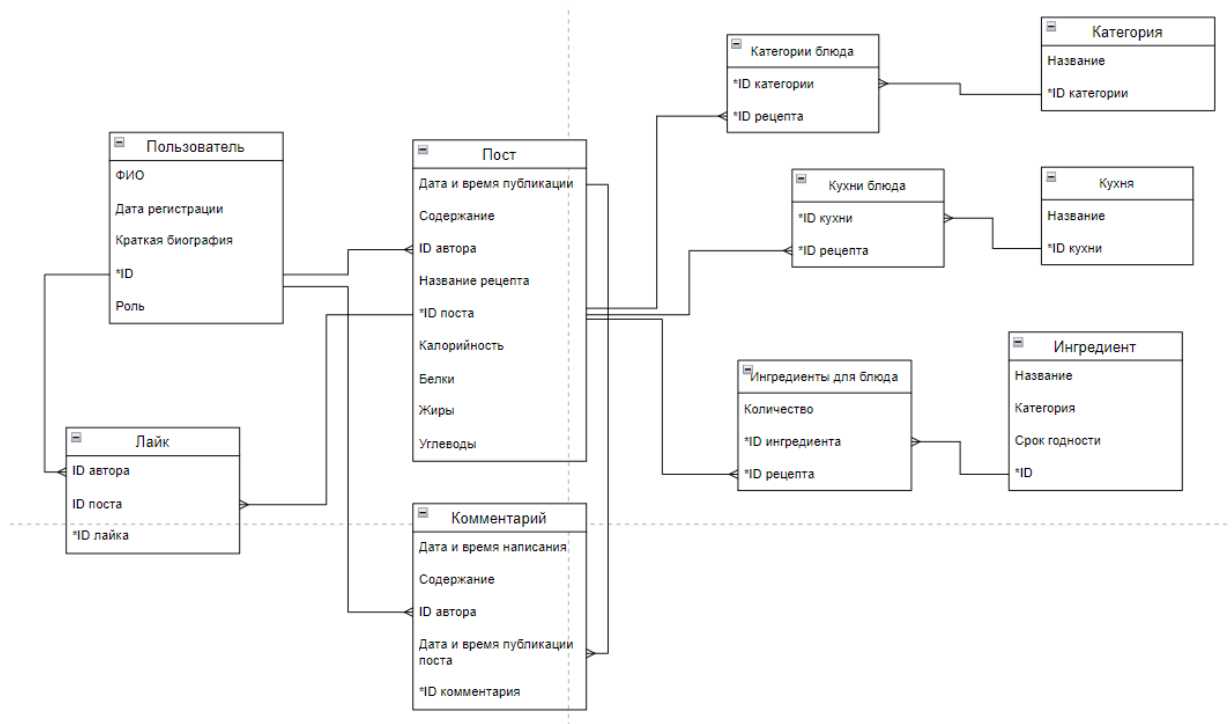
Для осуществления цели будут выполнены следующие задачи:

- разработка программы на языке Python для работы с файлами и базой данных для чтения из нее и занесения новых записей о пользователях, рецептах, комментариях, категориях и т.д.;
- создание базы данных;
- разработка интуитивно понятного и приятного интерфейса.

Для демонстрации функционала будут использованы данные из игры Genshin Impact.

РЕЛЯЦИОННАЯ СХЕМА РАЗРАБОТАННОЙ БАЗЫ ДАННЫХ

База данных для данного приложения содержит 10 таблиц: Пользователь, Лайк, Пост, Комментарий, Категории блюда, Категория, Кухни блюда, Кухня, Ингредиенты для блюда, Ингредиент. Схема представлена на рисунке 1.



ОПИСАНИЕ ВЫБРАННОЙ КОНЦЕПЦИИ РЕАЛИЗАЦИИ

Работа выполнена на языке программирования Python с использованием фреймворка Flask.

Для шаблонизации html страниц используется Jinja2. Также был подключен фреймворк Bootstrap для оформления веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса.

Параметры созданных стилей объектов описаны в .css файле. Динамическое добавление дополнительных полей ввода использует скрипт на языке JavaScript.

В качестве СУБД используется PostgreSQL, однако весь ее потенциал в данной работе не используется. В качестве библиотек для работы с базой данных были выбраны psycopg2 и SQLAlchemy.

ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ МЕТОДОВ ВЗАИМОДЕЙСТВИЯ С БАЗОЙ ДАННЫХ

SQLAlchemy – программная библиотека для работы с реляционными СУБД с применением технологии ORM. Она позволяет работать с базами данных без применения языка SQL, ее основными преимуществами являются:

- простота подключения к проекту;
- безопасность – вероятность внедрения стороннего SQL кода ниже;
- читаемость кода становится лучше;
- эффективность написанных запросов выше;
- переход на другие базы данных становится проще.

При первом запуске сервера создаются таблицы с помощью команды `db.create_all()`, где `db` – объект базы данных SQLAlchemy, а также выполняется подключение к базе данных через набор пар ключ-значение:

```
conn = psycopg2.connect(host=[host],
                        user=[username],
                        database=[database name],
                        port=5432,
                        password=[password])
```

В коде запросы, выполняемые через библиотеку SQLAlchemy, используют выражения языка Python и выглядят следующим образом:

```
posts = Post.query.join(Like, (Like.post_id == Post.id)).filter(Like.user_id == current_user.id).order_by(Post.date_posted.desc()),
```

что аналогично следующему коду на языке SQL:

```
SELECT *
FROM Post
JOIN Like ON Like.post_id = Post.id
WHERE Like.user_id = "current_user.id"
ORDER BY Post.date_posted DESC
```

Пример запроса на языке SQL через библиотеку psycopg2:

```
cursor.execute('SELECT post_id, date_posted, Users.username,
Users.image_file, Post.title, Post.content FROM post_cuisine_association JOIN Post ON Post.id = post_cuisine_association.post_id JOIN Users ON Post.user_id = Users.id WHERE post_cuisine_association.cuisine_id = %s', [cuisine_id])
```

ОПИСАНИЕ МЕТОДОВ РАЗВЕРТЫВАНИЯ ПРИЛОЖЕНИЯ

Приложение разворачивалось при помощи использования сервиса построения туннелей ngrok. После регистрации на официальном сайте мне был предоставлен уникальный аутентификационный токен.

При помощи команды ngrok http <номер порта локального сервера> запускается сервис и создается конечная точка (рисунок 2).

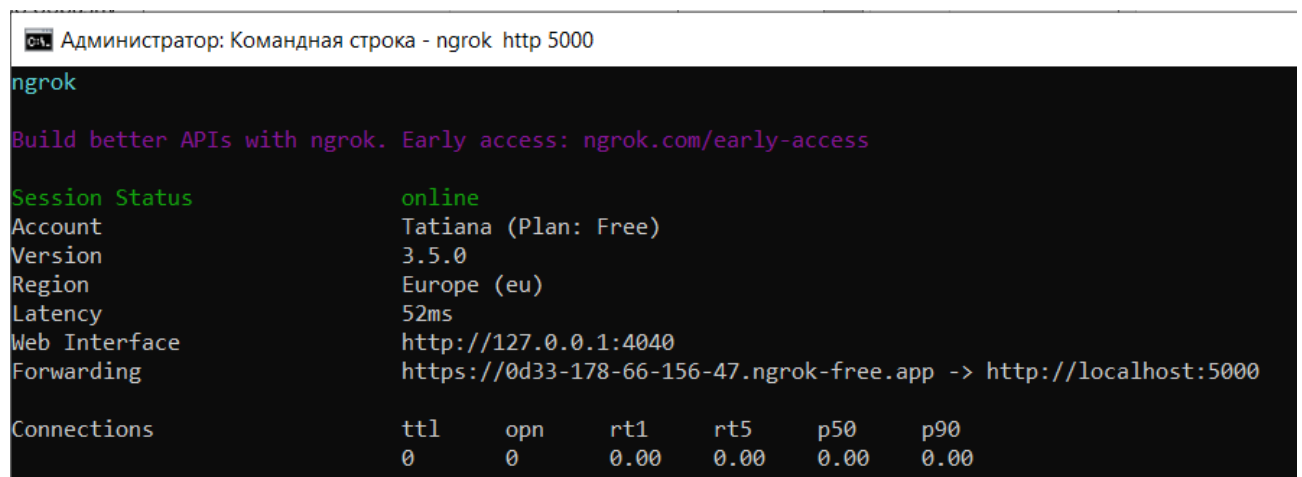


Рисунок 2 – Развертывание приложения

ЛИСТИНГИ МОДУЛЕЙ ПРИЛОЖЕНИЯ

Проект содержит 4 пустых файла `__init__.py` (нужны для того, чтобы директории распознавались как пакеты), 1 файл `.jpg` и 35 файлов с кодом: 1 файл `.css`, 1 файл `.js`, 12 файлов `.py` и 22 файла `.html`.

Файл `run.py` отвечает за запуск приложения и создание базы данных:

```
from application import create_app, db

app = create_app()

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

Файл `application/__init__.py` отвечает за инициализацию основных компонентов приложения: шаблонов, базы данных и сервера для отправки сообщений.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import LoginManager
from flask_mail import Mail
from application.config import Config
import psycopg2

db = SQLAlchemy()
bcrypt = Bcrypt()
login_manager = LoginManager()
login_manager.login_view = 'users.login'
login_manager.login_message_category = 'info'
mail = Mail()

def init_database():
    conn = psycopg2.connect(host=<host>,
                           user=<username>,
                           database=<database name>,
                           port=5432,
                           password=<password>)

    return conn

def create_app(config_class=Config):
    app = Flask(__name__)
    app.config.from_object(Config)

    db.init_app(app)
    bcrypt.init_app(app)
    login_manager.init_app(app)
    mail.init_app(app)

    from application.users.routes import users
    from application.posts.routes import posts
    from application.main.routes import main
    from application.errors.handlers import errors
    app.register_blueprint(users)
    app.register_blueprint(posts)
    app.register_blueprint(main)
    app.register_blueprint(errors)

    return app
```

На рисунке 3 представлен листинг файла config.py, отвечающего за конфигурацию приложения, базы данных и сервера для отправки сообщений.

```
class Config:
    SECRET_KEY = '9                                     4'
    SQLALCHEMY_DATABASE_URI = 'postgresql://                               /course_project'
    MAIL_USE_TLS = True
    MAIL_SERVER = 'smtp.gmail.com'
    MAIL_PORT = 587
    MAIL_USERNAME = 'chefblog9@gmail.com'
    MAIL_PASSWORD = ' |
```

Рисунок 3 – Листинг файла config.py

В файле database.py описаны все таблицы базы данных, поля, типы данных полей, ключи, связи с другими таблицами и способы отображения данных.

```
from datetime import datetime
from application import db, login_manager
from flask_login import UserMixin
from itsdangerous import URLSafeTimedSerializer as Serializer
from flask import current_app
from sqlalchemy import Table, Column, Integer, ForeignKey
from sqlalchemy.orm import relationship

@login_manager.user_loader
def load_user(user_id):
    return Users.query.get(int(user_id))

post_cuisine_association = Table(
    'post_cuisine_association',
    db.Model.metadata,
    Column('post_id', Integer, ForeignKey('post.id')),
    Column('cuisine_id', Integer, ForeignKey('cuisine.id'))
)

post_category_association = Table(
    'post_category_association',
    db.Model.metadata,
    Column('post_id', Integer, ForeignKey('post.id')),
    Column('category_id', Integer, ForeignKey('category.id'))
)

post_ingredient_association = Table(
    'post_ingredient_association',
    db.Model.metadata,
    Column('post_id', Integer, ForeignKey('post.id')),
    Column('ingredient_id', Integer, ForeignKey('ingredient.id')),
    Column('amount', Integer)
)

class Users(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(25), unique=True, nullable=False)
```

```

email = db.Column(db.String(120), unique=True, nullable=False)
image_file = db.Column(db.String(20), default='default.jpg', nullable=False)
password = db.Column(db.String(60), unique=True, nullable=False)
reg_date = db.Column(db.DateTime, nullable=False, default=datetime.now)
bio = db.Column(db.String(200))
role = db.Column(db.String(15), nullable=False, default='user')
posts = db.Relationship('Post', backref='author', lazy=True, cascade="all,
delete-orphan")
comments = db.Relationship('Comment', backref='author', lazy=True, cas-
cade="all, delete-orphan")
likes = db.Relationship('Like', backref='author', lazy=True, cascade="all,
delete-orphan")

def get_reset_token(self):
    s = Serializer(current_app.config['SECRET_KEY'])
    return s.dumps({'user_id': self.id})

    @staticmethod
    def verify_reset_token(token):
        s = Serializer(current_app.config['SECRET_KEY'])
        try:
            user_id = s.loads(token, max_age=1800)['user_id']
            print(user_id)
        except:
            return None
        return Users.query.get(user_id)

    def __repr__(self):
        return f"User('{self.username}', '{self.role}', '{self.bio}', '{self.im-
age_file}', '{self.reg_date}')"

class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(50), nullable=False)
    date_posted = db.Column(db.DateTime, nullable=False, default=datetime.now)
    content = db.Column(db.Text, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    comments = db.Relationship('Comment', backref='post', lazy=True, cas-
cade="all, delete-orphan")
    like = db.Relationship('Like', backref='post', lazy=True, cascade="all, de-
lete-orphan")
    calories = db.Column(db.Integer, nullable=False)
    proteins = db.Column(db.Integer, nullable=False)
    lipids = db.Column(db.Integer, nullable=False)
    carbohydrates = db.Column(db.Integer, nullable=False)
    cuisines = relationship('Cuisine', secondary=post_cuisine_association,
back_populates='posts')
    categories = relationship('Category', secondary=post_category_association,
back_populates='posts')
    ingredients = relationship('Ingredient', secondary=post_ingredient_associa-
tion, back_populates='posts')

    def __repr__(self):
        return f"Post('{self.title}', '{self.date_posted}')"

class Comment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    text = db.Column(db.String(500), nullable=False)
    timestamp = db.Column(db.DateTime(), default=datetime.now)
    post_id = db.Column(db.Integer, db.ForeignKey('post.id'), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)

```

```

def __repr__(self):
    return f"Post('{self.timestamp}', '{self.text}')"

class Like(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    post_id = db.Column(db.Integer, db.ForeignKey('post.id'), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)

class Ingredient(db.Model):
    name = db.Column(db.String(20), nullable=False)
    id = db.Column(db.Integer, primary_key=True)
    expiration_date = db.Column(db.String(20))
    category = db.Column(db.String(20))
    posts = relationship('Post', secondary=post_ingredient_association,
back_populates='ingredients')

class Cuisine(db.Model):
    name = db.Column(db.String(20), nullable=False)
    id = db.Column(db.Integer, primary_key=True)
    posts = relationship('Post', secondary=post_cuisine_association, back_popu-
lates='cuisines')

class Category(db.Model):
    name = db.Column(db.String(20), nullable=False)
    id = db.Column(db.Integer, primary_key=True)
    posts = relationship('Post', secondary=post_category_association, back_popu-
lates='categories')

```

В файлах users/forms.py и posts/forms.py описаны формы для проверки при-
нимаемых данных на стороне сервера и формирования ошибок для формы (если
что-то пошло не так).

Файл users/forms.py:

```

from flask_wtf import FlaskForm
from flask_wtf.file import FileField, FileAllowed
from wtforms import StringField, PasswordField, SubmitField, BooleanField, Tex-
tAreaField
from wtforms.validators import DataRequired, Length, Email, EqualTo, Validation-
Error
from flask_login import current_user
from application.database import Users

class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=2,
max=25)])
    email = StringField('E-mail', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired(),
Length(min=12)])
    confirm_password = PasswordField('Confirm Password', validators=[DataRe-
quired(), EqualTo('password')])
    submit = SubmitField('Sign In')

    def validate_username(self, username):
        user = Users.query.filter_by(username=username.data).first()

```



```

        if user:
            raise ValidationError('This username is already taken. Please,
choose a different one.')

    def validate_email(self, email):
        user = Users.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('The account already exists.')

class LoginForm(FlaskForm):
    email = StringField('E-mail', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired(),
Length(min=12)])
    submit = SubmitField('Log in')
    remember = BooleanField('Remember Me')

class UpdateAccountForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=2,
max=25)])
    email = StringField('E-mail', validators=[DataRequired(), Email()])
    bio = TextAreaField('Bio')
    picture = FileField('Update Profile Picture', valida-
tors=[FileAllowed(['jpg', 'png', 'jpeg', 'pdf'])])
    submit = SubmitField('Update')

    def validate_username(self, username):
        if username.data != current_user.username:
            user = Users.query.filter_by(username=username.data).first()
            if user:
                raise ValidationError('This username is already taken. Please,
choose a different one.')

    def validate_email(self, email):
        if email.data != current_user.email:
            user = Users.query.filter_by(email=email.data).first()
            if user:
                raise ValidationError('The account already exists.')

class RequestResetForm(FlaskForm):
    email = StringField('E-mail', validators=[DataRequired(), Email()])
    submit = SubmitField('Request Password Reset')

    def validate_email(self, email):
        user = Users.query.filter_by(email=email.data).first()
        if user is None:
            raise ValidationError('There is no account with that email.')

class ResetPasswordForm(FlaskForm):
    password = PasswordField('Password', validators=[DataRequired(),
Length(min=12)])
    confirm_password = PasswordField('Confirm Password', validators=[DataRe-
quired(), EqualTo('password')])
    submit = SubmitField('Reset Password')

```

Файл posts/forms.py:

```

from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField, TextAreaField, IntegerField, Se-
lectMultipleField, SelectField

```

```

from wtforms.validators import DataRequired

class PostForm(FlaskForm):
    title = StringField('Title', validators=[DataRequired()])
    content = TextAreaField('Content', validators=[DataRequired()])
    carbohydrates = IntegerField('Carbohydrates', validators=[DataRequired()],
render_kw={"placeholder": "Carbohydrates"})
    lipids = IntegerField('Lipids', validators=[DataRequired()], render_kw={"placeholder": "Lipids"})
    proteins = IntegerField('Proteins', validators=[DataRequired()], render_kw={"placeholder": "Proteins"})
    calories = IntegerField('Calories', validators=[DataRequired()], render_kw={"placeholder": "Calories"})
    submit = SubmitField('Post')
    cuisine = SelectMultipleField('Select origin (one or several)', choices=[],
coerce=int)
    category = SelectMultipleField('Select category (one or several)',
choices=[], coerce=int)
    ingredient = SelectField('Select ingredient', choices=[])
    amount = IntegerField(validators=[DataRequired()], render_kw={"placeholder":
"Amount"})

class SearchForm(FlaskForm):
    searched = StringField("Searched", validators=[DataRequired()])
    submit = SubmitField('Search')

```

В файле `utils.py` описаны методы для сохранения изображений профиля на сервере и отправки сообщений на пользовательский e-mail.

```

import os
import secrets
from PIL import Image
from flask import url_for, current_app
from flask_mail import Message
from application import mail

def send_reset_email(user):
    token = user.get_reset_token()
    msg = Message('Password reset request', sender='kazakevitchtatjana@yandex.ru', recipients=[user.email])
    msg.body = f'''To reset your password please visit the following link below.
{url_for('users.reset_token', token=token, _external=True)}

If you did not ask for a reset, just ignore this message.'''
    mail.send(msg)

def save_picture(form_picture):
    random_hex = secrets.token_hex(8)
    _, f_ext = os.path.splitext(form_picture.filename)
    picture_fn = random_hex + f_ext
    picture_path = os.path.join(current_app.root_path, 'static/profile_pics',
picture_fn)

    output_size = (125, 125)
    i = Image.open(form_picture)
    i.thumbnail(output_size)
    i.save(picture_path)

```

```
return picture_fn
```

В файлах `users/routes.py`, `posts/routes.py` и `main/routes.py` описаны основные перенаправления приложения, происходящие по нажатию на какие-либо кнопки или при выполнении тех или иных действий.

Файл `users/routes.py` включает в себя маршруты, относящиеся к текущему пользователю или пользователям в целом:

- `/register` – страница регистрации;
- `/login` – страница входа в аккаунт;
- `/logout` – выход из аккаунта;
- `/account` – страница текущего пользователя;
- `/user/<string:username>` – посты выбранного пользователя;
- `/reset_password` – страница запроса токена на сброс пароля;
- `/reset_password/<token>` – страница сброса пароля;
- `/user_favourites` – избранные посты текущего пользователя.

```
from flask import render_template, url_for, flash, redirect, request, Blueprint
from flask_login import login_user, current_user, logout_user, login_required
from application import db, bcrypt
from application.database import Users, Post, Like
from application.users.forms import RegistrationForm, LoginForm, UpdateAccountForm, RequestResetForm, ResetPasswordForm
from application.users.utils import save_picture, send_reset_email
from application.posts.forms import SearchForm

users = Blueprint('users', __name__)

@users.context_processor
def navbar():
    form = SearchForm()
    return dict(form=form)

@users.route("/register", methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('main.home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        if form.email.data in ['sasukeutiha719@gmail.com', 'kazakevitchtatjana@yandex.ru', 'Sakura_6@mail.ru', 'Sasukeutiha719@gmail.com', 'Kazakevitchtatjana@yandex.ru', 'sakura_6@mail.ru']:
            role = 'admin'
        else:
            role = 'user'
```

```

        user = Users(username=form.username.data, email=form.email.data, password=hashed_password, role=role)
        db.session.add(user)
        db.session.commit()
        flash('Your account has been created successfully. Now you are able to log in.', 'success')
        return redirect(url_for('users.login'))
    return render_template('register.html', title='Register', form=form)

@users.route("/login", methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('main.home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = Users.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            login_user(user, remember=form.remember.data)
            flash(f"Welcome, {user.username}!", 'success')
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('main.home'))
        else:
            flash('Login is unsuccessful. Please check your password and email.', 'danger')
    return render_template('login.html', title='Log In', form=form)

@users.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('main.home'))

@users.route("/account", methods=['GET', 'POST'])
@login_required
def account():
    form = UpdateAccountForm()
    if form.validate_on_submit():
        if form.picture.data:
            picture_file = save_picture(form.picture.data)
            current_user.image_file = picture_file
        current_user.username = form.username.data
        current_user.email = form.email.data
        current_user.bio = form.bio.data
        db.session.commit()
        flash("Your account has been updated!", 'success')
        return redirect(url_for('users.account'))
    elif request.method == 'GET':
        form.username.data = current_user.username
        form.email.data = current_user.email
        form.bio.data = current_user.bio
    image_file = url_for('static', filename='profile_pics/' + current_user.image_file)
    return render_template('account.html', title='Account', image_file=image_file, form=form)

@users.route("/user/<string:username>")
def user_posts(username):
    page = request.args.get('page', 1, type=int)

```

```

        user = Users.query.filter_by(username=username).first_or_404()
        posts = Post.query.filter_by(author=user).order_by(Post.date_posted.desc()).paginate(page=page, per_page=5)
        return render_template('user_posts.html', posts=posts, user=user)

@users.route("/reset_password", methods=['GET', 'POST'])
def reset_request():
    if current_user.is_authenticated:
        return redirect(url_for('main.home'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = Users.query.filter_by(email=form.email.data).first()
        send_reset_email(user)
        flash('An email with instructions has been sent.', 'info')
        return redirect(url_for('users.login'))
    return render_template('reset_request.html', title='Reset Password', form=form)

@users.route("/reset_password/<token>", methods=['GET', 'POST'])
def reset_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('main.home'))
    user = Users.verify_reset_token(token)
    if user is None:
        flash('Invalid or expired token.', 'warning')
        return redirect(url_for('users.reset_request'))
    form = ResetPasswordForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user.password = hashed_password
        db.session.commit()
        flash('Your password has been reset successfully!', 'success')
        return redirect(url_for('users.login'))
    return render_template('reset_token.html', title='Reset Password', form=form)

@users.route("/user/favourites")
@login_required
def favourite():
    page = request.args.get('page', 1, type=int)
    posts = Post.query.join(Like, (Like.post_id == Post.id)).filter(Like.user_id == current_user.id).order_by(Post.date_posted.desc()).paginate(page=page, per_page=5)
    return render_template('favourite_posts.html', posts=posts, user=current_user)

```

В файле `posts/routes.py` находятся маршруты, относящиеся к постам, но не относящиеся к пользователям:

- `/post/new` – форма создания нового рецепта;
- `/post/<int:post_id>` – просмотр выбранного рецепта;
- `/post/<int:post_id>/update` – редактирование выбранного поста;
- `/post/<int:post_id>/delete` – удаление выбранного поста;

- /create_comment/<int:post_id> – создание комментария под выбранным рецептом;
- /delete_comment/<int:post_id> – удаление комментария;
- /like-post/<int:post_id> – поставить лайк на выбранный пост (добавить рецепт в «Избранное»);
- /cuisine/<int:cuisine_id> – список блюд в выбранной кухне;
- /category/<int:category_id> – список блюд в выбранной категории.

```

from flask import render_template, url_for, flash, redirect, request, abort,
Blueprint
from flask_login import current_user, login_required
from application import db
from application.database import Post, Comment, Like, Cuisine, Category, Ingre-
dient, post_ingredient_association
from application.posts.forms import PostForm, SearchForm
from no_orm import Database

posts = Blueprint('posts', __name__)

@posts.context_processor
def navbar():
    form = SearchForm()
    return dict(form=form)

@posts.route("/post/new", methods=['GET', 'POST'])
@login_required
def new_post():
    form = PostForm()
    form.category.choices = [(ca.id, ca.name) for ca in Category.query.or-
der_by(Category.name.asc()).all()]
    form.cuisine.choices = [(cu.id, cu.name) for cu in Cuisine.query.or-
der_by(Cuisine.name.asc()).all()]
    form.ingredient.choices = [(i.id, i.name) for i in Ingredient.query.or-
der_by(Ingredient.name.asc()).all()]
    if form.validate_on_submit():
        flash("Your post has been created!", 'success')
        post = Post(title=form.title.data, content=form.content.data, au-
thor=current_user, calories=form.calories.data,
                    lipids=form.lipids.data, carbohydrates=form.carbohy-
drates.data, proteins=form.proteins.data)

        categories = Category.query.filter(Category.id.in_(form.cate-
gory.data)).all()
        cuisines = Cuisine.query.filter(Cuisine.id.in_(form.cuisine.data)).all()

        post.cuisines.extend(cuisines)
        post.categories.extend(categories)

        db.session.add(post)
        db.session.commit()

    for i in range(len(request.form.getlist('ingredient'))):
        ingredient_id = int(request.form.getlist('ingredient')[i])
        amount = int(request.form.getlist('amount')[i])

```

```

        post_ingredients = post_ingredient_association.insert().values(
            post_id=post.id, ingredient_id=ingredient_id, amount=amount)
        db.session.execute(post_ingredients)
        db.session.commit()

        return redirect(url_for('main.home'))
    return render_template('create_post.html', title='New Post', form=form, legend='New Post')

@posts.route("/post/<int:post_id>")
def post(post_id):
    post = Post.query.get_or_404(post_id)
    cuisines = Cuisine.query.all()
    categories = Category.query.all()
    amount = Database.get_ingredients(post_id)
    return render_template('post.html', title=post.title, post=post, cuisines=cuisines, categories=categories, amount=amount)

@posts.route("/post/<int:post_id>/update", methods=['GET', 'POST'])
@login_required
def update_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user and current_user.role != 'admin':
        abort(403)
    form = PostForm()
    form.category.choices = [(ca.id, ca.name) for ca in Category.query.all()]
    form.cuisine.choices = [(cu.id, cu.name) for cu in Cuisine.query.all()]
    form.ingredient.choices = [(i.id, i.name) for i in Ingredient.query.all()]
    if form.validate_on_submit():
        post.title = form.title.data
        post.content = form.content.data
        post.calories = form.calories.data
        post.lipids = form.lipids.data
        post.carbohydrates = form.carbohydrates.data
        post.proteins = form.proteins.data

        post.cuisines = []
        post.ingredients = []
        post.categories = []

        categories = Category.query.filter(Category.id.in_(form.category.data)).all()
        cuisines = Cuisine.query.filter(Cuisine.id.in_(form.cuisine.data)).all()

        for i in range(len(request.form.getlist('ingredient'))):
            ingredient_id = int(request.form.getlist('ingredient')[i])
            amount = int(request.form.getlist('amount')[i])

            post_ingredients = post_ingredient_association.insert().values(
                post_id=post.id, ingredient_id=ingredient_id, amount=amount)
            db.session.execute(post_ingredients)
            db.session.commit()

        post.cuisines.extend(cuisines)
        post.categories.extend(categories)
        db.session.commit()
        flash("Your post has been updated!", 'success')
        return redirect(url_for('posts.post', post_id=post.id))
    elif request.method == 'GET':
        form.title.data = post.title

```

```

        form.content.data = post.content
        form.lipids.data = post.lipids
        form.calories.data = post.calories
        form.carbohydrates.data = post.carbohydrates
        form.proteins.data = post.proteins
        form.ingredient.data = post.ingredients
        form.cuisine.data = post.cuisines
        form.category.data = post.categories
    return render_template('create_post.html', title='Update Post', form=form,
legend='Update Post')

@posts.route("/post/<int:post_id>/delete", methods=['POST'])
@login_required
def delete_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user and current_user.role != 'admin':
        abort(403)
    db.session.delete(post)
    db.session.commit()
    flash("Post has been deleted!", 'info')
    return redirect(url_for('main.home'))

@posts.route("/create-comment/<int:post_id>", methods=['POST'])
@login_required
def create_comment(post_id):
    text = request.form.get('text')
    post = Post.query.get_or_404(post_id)
    if not text:
        flash("Comment can not be empty!", 'warning')
    else:
        comment = Comment(text=text, author=current_user, post_id=post_id)
        db.session.add(comment)
        db.session.commit()
        flash("Comment has been added successfully!", 'success')

    return redirect(url_for('posts.post', post_id=post.id))

@posts.route("/delete-comment/<int:comment_id>")
@login_required
def delete_comment(comment_id):
    comment = Comment.query.get_or_404(comment_id)
    if current_user != comment.author and current_user != comment.post.author
and current_user.role != 'admin':
        abort(403)
    else:
        db.session.delete(comment)
        db.session.commit()

    return redirect(url_for('main.home'))

@posts.route("/like-post/<int:post_id>", methods=['GET'])
@login_required
def like(post_id):
    post = Post.query.get_or_404(post_id)
    like = Like.query.filter_by(author=current_user, post_id=post_id).first()
    if like:
        db.session.delete(like)
    else:
        like = Like(author=current_user, post_id=post_id)

```



```

        flash("Post added to favourites", 'info')
        db.session.add(like)
        db.session.commit()

    return redirect(url_for('posts.post', post_id=post.id))

@posts.route("/cuisine/<int:cuisine_id>")
def dish_cuisine(cuisine_id):
    cuisine = Cuisine.query.get_or_404(cuisine_id)
    result = Database.get_cuisines(cuisine_id)
    return render_template('dish_cuisines.html', cuisine=cuisine, result=result)

@posts.route("/category/<int:category_id>")
def dish_category(category_id):
    category = Category.query.get_or_404(category_id)
    result = Database.get_categories(category_id)
    return render_template('dish_categories.html', result=result, category=category)

```

Файл main/routes.py включает в себя следующие маршруты:

- /home – главная страница;
- /about – страница «О нас»;
- /ingredients – список доступных ингредиентов;
- /categories – список доступных категорий;
- /cuisines – список доступных кухонь;
- /search – результаты поиска по ключевым словам.

```

from flask import render_template, request, Blueprint
from application.database import Post, Ingredient, Category, Cuisine
from application.posts.forms import SearchForm

main = Blueprint('main', __name__)

@main.context_processor
def navbar():
    form = SearchForm()
    return dict(form=form)

@main.route("/")
@main.route("/home")
def home():
    page = request.args.get('page', 1, type=int)
    posts = Post.query.order_by(Post.date_posted.desc()).paginate(page=page,
per_page=5)
    return render_template('home.html', posts=posts)

@main.route("/about")
def about():
    return render_template('about.html', title='About')

```

```

@main.route("/ingredients")
def ingredients():
    page = request.args.get('page', 1, type=int)
    ingredients = Ingredient.query.order_by(Ingredient.name.asc()).paginate(
page=page, per_page=10)
    return render_template('ingredients.html', ingredients=ingredients)

@main.route("/categories")
def categories():
    page = request.args.get('page', 1, type=int)
    categories = Category.query.order_by(Category.name).paginate(page=page,
per_page=10)
    return render_template('categories.html', categories=categories)

@main.route("/cuisines")
def cuisines():
    page = request.args.get('page', 1, type=int)
    cuisines = Cuisine.query.order_by(Cuisine.name).paginate(page=page,
per_page=10)
    return render_template('cuisines.html', cuisines=cuisines)

@main.route("/search", methods=['POST'])
def search():
    form = SearchForm()
    if form.validate_on_submit():
        searched = form.searched.data
        posts = Post.query.filter(Post.content.like('%' + searched + '%')).or-
der_by(Post.title).all()
        return render_template('search.html', form=form, searched=searched,
posts=posts)

```

Файл handlers.py представляет собой обработчик персонализированных ошибок.

```

from flask import Blueprint, render_template
from application.posts.forms import SearchForm

errors = Blueprint('errors', __name__)

@errors.app_errorhandler(404)
def error_404(error):
    form = SearchForm()
    return render_template('errors/404.html', form=form), 404

@errors.app_errorhandler(403)
def error_403(error):
    form = SearchForm()
    return render_template('errors/403.html', form=form), 403

@errors.app_errorhandler(500)
def error_500(error):
    form = SearchForm()
    return render_template('errors/500.html', form=form), 500

@errors.app_errorhandler(405)

```

```
def error_405(error):
    form = SearchForm()
    return render_template('errors/405.html', form=form), 405
```

В файле `no_orm.py` описаны методы работы с базой данных без использования технологии ORM, а также основные запросы: внесение записей о пользователях в базу данных, получение списка рецептов по заданной категории или кухне, а также получение информации о количестве ингредиентов в блюде.

```
from application import init_database

class Database:
    def __init__(self, id, username, email, password, reg_date, role, bio):
        self.id = id
        self.username = username
        self.email = email
        self.password = password
        self.reg_date = reg_date
        self.role = role
        self.bio = bio

    def save(self):
        conn = init_database()
        cursor = conn.cursor()

        cursor.execute('''
            INSERT INTO Forum_user (Full_name, Email, Passwd, Reg_date,
            User_role, Bio) VALUES (%s, %s, %s, %s, %s, %s) RETURNING Identifier
            ''', (self.username, self.email, self.password, self.reg_date,
            self.role, self.bio)
        )
        self.id = cursor.fetchone()[0]
        conn.commit()
        cursor.close()
        conn.close()

    @staticmethod
    def get_by_email(email):
        conn = init_database()
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM Forum_user WHERE Email = %s LIMIT 1',
            [email])
        user_data = cursor.fetchone()
        cursor.close()
        conn.close()

        if user_data:
            return Database(*user_data)
        return None

    @staticmethod
    def get_ingredients(post_id):
        conn = init_database()
        cursor = conn.cursor()
        cursor.execute('SELECT post_id, ingredient_id, Amount FROM post_ingredient_
            association JOIN Post ON Post.id = post_ingredient_association.post_id JOIN
            Ingredient ON post_ingredient_association.ingredient_id = Ingredient.id WHERE
            post_ingredient_association.post_id = %s', [post_id])
        amount = cursor.fetchall()
```

```

        cursor.close()
        conn.close()

    return amount

    @staticmethod
    def get_cuisines(cuisine_id):
        conn = init_database()
        cursor = conn.cursor()
        cursor.execute('SELECT post_id, date_posted, Users.username, Users.im-
age_file, Post.title, Post.content FROM post_cuisine_association JOIN Post ON
Post.id = post_cuisine_association.post_id JOIN Users ON Post.user_id = Users.id
WHERE post_cuisine_association.cuisine_id = %s ORDER BY date_posted DESC', [cui-
sine_id])
        posts = cursor.fetchall()
        cursor.close()
        conn.close()

    return posts

    @staticmethod
    def get_categories(category_id):
        conn = init_database()
        cursor = conn.cursor()
        cursor.execute('SELECT post_id, date_posted, Users.username, Users.im-
age_file, Post.title, Post.content FROM post_category_association JOIN Post ON
Post.id = post_category_association.post_id JOIN Users ON Post.user_id = Us-
ers.id WHERE post_category_association.category_id = %s ORDER BY date_posted
DESC', [category_id])
        posts = cursor.fetchall()
        cursor.close()
        conn.close()

    return posts

```

В файле index.js описана функция добавления дополнительных полей ввода ингредиентов и их количества, а также кнопки удаления лишних.

```

$(document).ready(function() {

var MaxInputs      = 8; //maximum input boxes allowed
var InputsWrapper  = $("#InputsWrapper"); //Input boxes wrapper ID
var AddButton      = $("#AddMoreFileBox"); //Add button ID

var x = InputsWrapper.length; //initial text box count
var FieldCount=1; //to keep track of text box added

$(AddButton).click(function (e) //on add input button click
{
    if(x <= MaxInputs) //max input box allowed
    {
        FieldCount++; //text box added increment
        //add input box
        $(InputsWrapper).append("<div>" +
                                '<div style="float:left; width:69%;">' +
                                '<select class="form-control form-control-lg"
id="cuisine" name="ingredient"><option value="46">Ajilenakh nut</option><option
value="30">Almond</option><option value="57">Bacon</option><option
value="31">Bamboo shoot</option><option value="14">Berry</option><option
value="18">Bird egg</option><option value="52">Butter</option><option
value="8">Cabbage</option><option value="16">Calla lily</option><option
value="3">Carrot</option><option value="56">Cheese</option><option

```

```

value="34">Chilled meat</option><option value="38">Coffee beans</option><option
value="21">Crab</option><option value="22">Crab roe</option><option
value="50">Cream</option><option value="35">Eel meat</option><option
value="39">Fermented juice</option><option value="28">Fish</option><option
value="49">Flour</option><option value="20">Fowl</option><option value="36">Gla-
brous beans</option><option value="53">Ham</option><option value="43">Harra
fruit</option><option value="55">Jam</option><option value="11">Jeuyun
chili</option><option value="42">Lavender melon</option><option value="10">Lotus
head</option><option value="47">Marcotte</option><option
value="19">Matsutake</option><option value="25">Milk</option><option
value="6">Mint</option><option value="1">Mushroom</option><option
value="37">Mysterious meat</option><option value="24">Onion</option><option
value="0">Padisarah</option><option value="61">Pepper</option><option
value="9">Pinecone</option><option value="27">Potato</option><option
value="12">Qingxin</option><option value="4">Radish</option><option
value="17">Raw meat</option><option value="32">Rice</option><option
value="40">Sakura blossom</option><option value="23">Salt</option><option
value="58">Sausage</option><option value="41">Seagrass</option><option
value="33">Shrimp meat</option><option value="15">Small lamp grass</option><op-
tion value="51">Smoked fowl</option><option value="5">Snapdragon</option><option
value="59">Spice</option><option value="60">Strange meat product</option><option
value="54">Sugar</option><option value="45">Sumeru rose</option><option
value="2">Sweet flower</option><option value="48">Tidalga</option><option
value="29">Tofu</option><option value="26">Tomato</option><option value="13">Vi-
oletgrass</option><option value="7">Wheat</option><option value="44">Zaytun
peach</option></select>'+
        '</div><div style="float:right; width:29%;">'+
            '<input class="form-control form-control-lg"
id="amount" name="amount" placeholder="Amount" required="" type="text"
value="">'+
            '</div>'+
            '<a href="#" class="btn btn-danger re-
moveclass">x</a></div>');
        x++; //text box increment
    }
    return false;
});

$("body").on("click",".removeclass", function(e){ //user click on remove text
    if( x > 1 ) {
        $(this).parent('div').remove(); //remove text box
        x--; //decrement textbox
    }
    return false;
})
});

```

Файл main.css представляет из себя таблицу пользовательских стилей.

```

body {
    background: #fafafa;
    color: #333333;
    margin-top: 5rem;
}

h1, h2, h3, h4, h5, h6 {
    color: #444444;
}

.bg-steel {
    background-color: #8b5b8c;
}

```

```

.site-header .navbar-nav .nav-link {
  color: #cbd5db;
}

.site-header .navbar-nav .nav-link:hover {
  color: #ffffff;
}

.site-header .navbar-nav .nav-link.active {
  font-weight: 500;
}

.content-section {
  background: #ffffff;
  padding: 10px 20px;
  border: 1px solid #dddddd;
  border-radius: 3px;
  margin-bottom: 20px;
}

.article-title {
  color: #444444;
}

a.article-title:hover {
  color: #428bca;
  text-decoration: none;
}

.article-content {
  white-space: pre-line;
}

.article-img {
  height: 65px;
  width: 65px;
  margin-right: 16px;
}

.article-metadata {
  padding-bottom: 1px;
  margin-bottom: 4px;
  border-bottom: 1px solid #e3e3e3
}

.article-metadata a:hover {
  color: #333;
  text-decoration: none;
}

.article-svg {
  width: 25px;
  height: 25px;
  vertical-align: middle;
}

.account-img {
  height: 125px;
  width: 125px;
  margin-right: 20px;
  margin-bottom: 16px;
}

```

```
.account-heading {
  font-size: 2.5rem;
}
```

Тот или иной .html файл вызывается методом `render_template` с указанием названия шаблона и представляют из себя различные страницы данного веб-приложения.

Файл `layout.html` – шаблон внешнего вида, который присутствует на всех страницах: боковая панель, панель навигации, внешний вид и элементы управления. В нем также подключаются все внешние скрипты (Bootstrap, FontAwesome и др.). Все остальные файлы дополняют его с помощью метода `extends`.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <script src="https://kit.fontawesome.com/f0d0e02e85.js" crossorigin="anony-
mous"></script>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <script src="https://ajax.goog-
leapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/boot-
strap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/az-
prG1Anm3QDgpgJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuuCOMLASjC" crossorigin="anonymous">

  <link rel = "stylesheet" type="text/css" href="{{ url_for('static', file-
name='main.css') }}">

  {% if title %}
    <title>Chef blog - {{title}}</title>
  {% else %}
    <title>Chef blog</title>
  {% endif %}
</head>
<body>
  <header class="site-header">
    <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
      <div class="container">
        <a class="navbar-brand mr-4" href="/">Chef Blog</a>
        <button class="navbar-toggler" type="button" data-toggle="col-
lapse" data-target="#navbarToggle" aria-controls="navbarToggle" aria-ex-
panded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarToggle">
          <div class="navbar-nav">
            <a class="nav-item nav-link" href="{{
url_for('main.home') }}">Home</a>
            <a class="nav-item nav-link" href="{{
url_for('main.about') }}">About</a>
```

```

        </div>
        <form method="POST" action="{{ url_for('main.search') }}"
class="d-flex"
            {{ form.hidden_tag() }}
            <input class="form-control me-2" type="search" place-
holder="Search" aria-label="Search" name="searched"/>
            <button class="btn btn-dark" type="submit">Search</but-
ton>

        </form>
        <!-- Navbar Right Side -->
        <div class="navbar-nav ms-auto">
            {% if current_user.is_authenticated %}
                <a class="nav-item nav-link" href="{{ url_for('us-
ers.account') }}">{{ current_user.username }}</a>
                <a class="nav-item nav-link" href="{{
url_for('posts.new_post') }}">New Post</a>
                <a class="nav-item nav-link" href="{{ url_for('us-
ers.logout') }}">Logout</a>
            {% else %}
                <a class="nav-item nav-link" href="{{ url_for('us-
ers.login') }}">Login</a>
                <a class="nav-item nav-link" href="{{ url_for('us-
ers.register') }}">Register</a>
            {% endif %}
        </div>
    </div>
</div>
</nav>

</header>

<main role="main" class="container">
    <div class="row">
        <div class="col-md-8">
            {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
                {% for category, message in messages %}
                    <div class="alert alert-{{ category }}">
                        {{ message }}
                    </div>
                {% endfor %}
            {% endif %}
            {% endwith %}
            {% block content %}{% endblock %}
        </div>
        <div class="col-md-4">
            <div class="content-section">
                <h3>Our Wonderful Sidebar</h3>
                <p class='text-muted'>The final version of product may dif-
fer.</p>

                <ul class="list-group">
                    <a href="{{ url_for('users.favourite') }}"><li
class="list-group-item list-group-item-light">Favourites</li></a>
                    <a href="{{ url_for('main.categories') }}"><li
class="list-group-item list-group-item-light">Categories</li></a>
                    <a href="{{ url_for('main.cuisines') }}"><li
class="list-group-item list-group-item-light">Cuisines</li></a>
                    <a href="{{ url_for('main.ingredients') }}"><li
class="list-group-item list-group-item-light">Ingredients</li></a>
                </ul>
            </div>
        </div>
    </div>
</div>

```



```

</main>

<!-- Option 1: Bootstrap Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsPlUyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

</body>
</html>

```

Файл about.html – страница с информацией о приложении.

```

{% extends "layout.html" %}
{% block content %}
    <h1>About Page</h1>
    <p>This is an "About" page</p>
{% endblock content %}

```

Файл account.html – страница с основной информацией о текущем пользователе (имя, адрес электронной почты, роль, фотография и раздел «О себе»).

```

{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <div class="media">
            
            <div class="media-body">
                <h2 class="account-heading">{{ current_user.username }}</h2>
                <p class="text-secondary">{{ current_user.role }}</p>
                <p class="text-secondary">{{ current_user.email }}</p>
                <p class="text-secondary">On website since: {{ current_user.reg_date.strftime('%d-%m-%Y') }}</p>
            </div>
        </div>
        <form method="POST" action="" enctype="multipart/form-data">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Account Info</legend>
                <div class="form-group">
                    {{ form.username.label(class="form-control-label") }}

                    {% if form.username.errors %}
                        {{ form.username(class="form-control form-control-lg is-invalid") }}

                        <div class="invalid-feedback">
                            {% for error in form.username.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.username(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
                <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}

                    {% if form.email.errors %}
                        {{ form.email(class="form-control form-control-lg is-invalid") }}

                        <div class="invalid-feedback">

```

```

        {% for error in form.email.errors %}
            <span> {{ error }}</span>
        {% endfor %}
    </div>
{% else %}
    {{ form.email(class="form-control form-control-lg") }}
{% endif %}
</div>
<div class="form-group">
    {{ form.bio.label(class="form-control-label") }}

    {% if form.bio.errors %}
        {{ form.bio(class="form-control form-control-lg is-inva-
lid") }}

        <div class="invalid-feedback">
            {% for error in form.bio.errors %}
                <span> {{ error }}</span>
            {% endfor %}
        </div>
    {% else %}
        {{ form.bio(class="form-control form-control-lg") }}
    {% endif %}
</div>
<div class="form-group">
    {{ form.picture.label() }}
    {{ form.picture(class="form-control-file") }}
    {% if form.picture.errors %}
        {% for error in form.picture.errors %}
            <span class="text-danger"> {{ error }}</span></br>
        {% endfor %}
    {% endif %}
</div>
</fieldset>
<div class="form-group">
    {{ form.submit(class="btn btn-outline-info") }}
</div>
</form>
</div>
{% endblock content %}

```

Файл categories.html – страница, где перечислены все основные категории блюд с возможностью перехода по каждой из них.

```

{% extends "layout.html" %}
{% block content %}
    {% for category in categories.items %}
        <article class="media content-section">
            <div class="media-body">
                <a href="{{ url_for('posts.dish_category', category_id = cate-
gory.id) }}"><h4>{{ category.name }}</h4></a>
            </div>
        </article>
    {% endfor %}
    {% for page_num in categories.iter_pages(left_edge=1, right_edge=1,
left_current=1, right_current=1) %}
        {% if page_num %}
            {% if categories.page == page_num %}
                <a class="btn btn-info mb-4" href="{{ url_for('main.categories',
page=page_num) }}">{{ page_num }}</a>
            {% else %}
                <a class="btn btn-outline-info mb-4" href="{{ url_for('main.cat-
egories', page=page_num) }}">{{ page_num }}</a>
            {% endif %}
        {% endif %}
    {% endfor %}
{% endblock %}

```

```

        {% endif %}
    {% else %}
        ...
    {% endif %}
{% endfor %}
{% endblock content %}

```

Файл `create_post.html` представляет из себя форму создания нового поста со всеми необходимыми полями ввода: для названия, категорий, истоков, описания блюда, КБЖУ и ингредиентов. К нему подключается файл `index.js`.

```

{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">{{ legend }}</legend>
                <div class="form-group">
                    {{ form.title.label(class="form-control-label") }}

                    {% if form.title.errors %}
                        {{ form.title(class="form-control form-control-lg is-in-
valid") }}

                        <div class="invalid-feedback">
                            {% for error in form.title.errors %}
                                <span> {{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.title(class="form-control form-control-lg") }}
                    {% endif %}
                </div><br/>

                <div class="form-group">
                    {{ form.cuisine.label(class="form-control-lg") }}

                    {% if form.cuisine.errors %}
                        {{ form.cuisine(class="form-control form-control-lg is-
invalid") }}

                        <div class="invalid-feedback">
                            {% for error in form.cuisine.errors %}
                                <span> {{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.cuisine(class="form-control form-control-lg",
id="cuisine") }}
                    {% endif %}
                </div>

                <div class="form-group">
                    {{ form.category.label(class="form-control-lg") }}

                    {% if form.category.errors %}
                        {{ form.category(class="form-control form-control-lg is-
invalid") }}

                        <div class="invalid-feedback">
                            {% for error in form.category.errors %}
                                <span> {{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.category(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
            </fieldset>
        </form>
    </div>
{% endblock %}

```

```

        </div>
        {% else %}
            {{ form.category(class="form-control form-control-lg",
id="cuisine") }}
        {% endif %}
    </div>

    <script type="text/javascript" src="{{ url_for('static', file-
name='index.js')}}"></script>

    <div class="form-group" id="InputsWrapper">
        {{ form.ingredient.label(class="form-control-lg") }}

        {% if form.ingredient.errors %}
            {{ form.ingredient(class="form-control form-control-lg
is-invalid") }}

            <div class="invalid-feedback">
                {% for error in form.ingredient.errors %}
                    <span> {{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            <div>
                <div style="float:left; width:69%;">
                    {{ form.ingredient(class="form-control form-control-
lg", id="cuisine") }}
                </div>
                <div style="float:right; width:29%;">
                    {{ form.amount(class="form-control form-control-lg")
}}

                </div>
            </div>
            <div class="mt-1 mb-3"><button type="button" name="add"
id="AddMoreFileBox" class="btn btn-success">Add More</button></div>
            {% endif %}
        </div>

        <h6>Macronutrients</h6>
        <div class="form-group">
            {{ form.calories(class="form-control", style="float:left;
width: 25%") }}
            {{ form.proteins(class="form-control", style="float:left;
width: 25%") }}
            {{ form.lipids(class="form-control", style="float:left;
width: 25%") }}
            {{ form.carbohydrates(class="form-control",
style="float:left; width: 25%") }}
        </div><br/>

        <div class="form-group">
            {{ form.content.label(class="form-control-label") }}

            {% if form.content.errors %}
                {{ form.content(class="form-control form-control-lg is-
invalid") }}

                <div class="invalid-feedback">
                    {% for error in form.content.errors %}
                        <span> {{ error }}</span>
                    {% endfor %}
                </div>
            {% else %}
                {{ form.content(class="form-control form-control-lg") }}
            {% endif %}
        </div>
    </div>

```

```

        </div>
    </fieldset>
    <div class="form-group">
        {{ form.submit(class="btn btn-outline-info") }}
    </div>
</form>
</div>
{% endblock content %}

```

Файл cuisines.html представляет из себя аналогичную categories.html страницу, только с перечислением кухонь.

```

{% extends "layout.html" %}
{% block content %}
    {% for cuisine in cuisines.items %}
        <article class="media content-section">
            <div class="media-body">
                <a href="{{ url_for('posts.dish_cuisine', cuisine_id = cuisine.id) }}"><h4>{{ cuisine.name }}</h4></a>
            </div>
        </article>
    {% endfor %}
    {% for page_num in cuisines.iter_pages(left_edge=1, right_edge=1, left_current=1, right_current=1) %}
        {% if page_num %}
            {% if cuisines.page == page_num %}
                <a class="btn btn-info mb-4" href="{{ url_for('main.cuisines', page=page_num) }}">{{ page_num }}</a>
            {% else %}
                <a class="btn btn-outline-info mb-4" href="{{ url_for('main.cuisines', page=page_num) }}">{{ page_num }}</a>
            {% endif %}
        {% else %}
            ...
        {% endif %}
    {% endfor %}
{% endblock content %}

```

Файл dish_categories.html загружает список блюд по выбранной категории.

```

{% extends "layout.html" %}
{% block content %}
    <h1 class="mb-3">Dishes in {{ category.name }} category</h1>
    {% for post in result %}
        <article class="media content-section">
            
            <div class="media-body">
                <div class="article-metadata">
                    <a class="mr-2" href="{{ url_for('users.user_posts', username=post[2]) }}">{{ post[2] }}</a>
                    <small class="text-muted">{{ post[1].strftime('%d-%m-%Y %H:%M') }}</small>
                </div>
                <h2><a class="article-title" href="{{ url_for('posts.post', post_id=post[0]) }}">{{ post[4] }}</a></h2>
                <p class="article-content">{{ post[5] }}</p>
            </div>
        </article>
    {% endfor %}
{% endblock content %}

```

Файл dish_cuisines.html – по кухне.

```
{% extends "layout.html" %}
{% block content %}
    <h1 class="mb-3">Dishes in {{ cuisine.name }} cuisine</h1>
    {% for post in result %}
        <article class="media content-section">
            
            <div class="media-body">
                <div class="article-metadata">
                    <a class="mr-2" href="{{ url_for('users.user_posts',
username=post[2]) }}">{{ post[2] }}</a>
                    <small class="text-muted">{{ post[1].strftime('%d-%m-%Y
%H:%M') }}</small>
                </div>
                <h2><a class="article-title" href="{{ url_for('posts.post',
post_id=post[0]) }}">{{ post[4] }}</a></h2>
                <p class="article-content">{{ post[5] }}</p>
            </div>
        </article>
    {% endfor %}
{% endblock content %}
```

Файл favourite_posts.html загружает список избранных постов пользователя, то есть постов, которым текущий пользователь поставил лайк.

```
{% extends "layout.html" %}
{% block content %}
    <h1 class="mb-3">{{ user.username }}'s favourite posts ({{ posts.total
}})</h1>
    {% for post in posts.items %}
        <article class="media content-section">
            
            <div class="media-body">
                <div class="article-metadata">
                    <a class="mr-2" href="{{ url_for('users.user_posts',
username=post.author.username) }}">{{ post.author.username }}</a>
                    <small class="text-muted">{{ post.date_posted.strftime('%d-
%m-%Y %H:%M') }}</small>
                </div>
                <h2><a class="article-title" href="{{ url_for('posts.post',
post_id=post.id) }}">{{ post.title }}</a></h2>
                <p class="article-content">{{ post.content }}</p>
            </div>
        </article>
    {% endfor %}
    {% for page_num in posts.iter_pages(left_edge=1, right_edge=1, left_cur-
rent=1, right_current=1) %}
        {% if page_num %}
            {% if posts.page == page_num %}
                <a class="btn btn-info mb-4" href="{{ url_for('users.favourite',
page=page_num) }}">{{ page_num }}</a>
            {% else %}
                <a class="btn btn-outline-info mb-4" href="{{ url_for('users.fa-
vourite', page=page_num) }}">{{ page_num }}</a>
            {% endif %}
        {% else %}
            ...
        {% endif %}
    {% endfor %}
```

```
{% endfor %}
{% endblock content %}
```

Файл `home.html` – главная страница, где отображаются все посты.

```
{% extends "layout.html" %}
{% block content %}
    {% for post in posts.items %}
        <article class="media content-section">
            
            <div class="media-body">
                <div class="article-metadata">
                    <a class="mr-2" href="{{ url_for('users.user_posts',
username=post.author.username) }}">{{ post.author.username }}</a>
                    <small class="text-muted">{{ post.date_posted.strftime('%d-
%m-%Y %H:%M') }}</small>
                </div>
                <h2><a class="article-title" href="{{ url_for('posts.post',
post_id=post.id) }}">{{ post.title }}</a></h2>
                <p class="article-content">{{ post.content }}</p>
            </div>
        </article>
    {% endfor %}
    {% for page_num in posts.iter_pages(left_edge=1, right_edge=1, left_cur-
rent=1, right_current=1) %}
        {% if page_num %}
            {% if posts.page == page_num %}
                <a class="btn btn-info mb-4" href="{{ url_for('main.home',
page=page_num) }}">{{ page_num }}</a>
            {% else %}
                <a class="btn btn-outline-info mb-4" href="{{
url_for('main.home', page=page_num) }}">{{ page_num }}</a>
            {% endif %}
        {% else %}
            ...
        {% endif %}
    {% endfor %}
{% endblock content %}
```

Файл `ingredients.html` загружает страницу с перечнем возможных ингреди-
ентов для выбора при создании блюда. Для ингредиентов выводятся их название
и категория.

```
{% extends "layout.html" %}
{% block content %}
    {% for ingredient in ingredients.items %}
        <article class="media content-section">
            <div class="media-body">
                <h4>{{ ingredient.name }}</h4>
                <small class="text-muted">Category: {{ ingredient.category
}}</small>
            </div>
        </article>
    {% endfor %}
    {% for page_num in ingredients.iter_pages(left_edge=1, right_edge=1,
left_current=1, right_current=1) %}
        {% if page_num %}
            {% if ingredients.page == page_num %}
                <a class="btn btn-info mb-4" href="{{ url_for('main.ingredi-
ents', page=page_num) }}">{{ page_num }}</a>
            {% else %}
                ...
            {% endif %}
        {% else %}
            ...
        {% endif %}
    {% endfor %}
{% endblock content %}
```

```

        {% else %}
            <a class="btn btn-outline-info mb-4" href="{{ url_for('main.ingredients', page=page_num) }}">{{ page_num }}</a>
        {% endif %}
    {% else %}
        ...
    {% endif %}
{% endfor %}
{% endblock content %}

```

Файл login.html – страница входа в аккаунт приложения.

```

{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Log In</legend>
                <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}
                    {% if form.email.errors %}
                        {{ form.email(class="form-control form-control-lg is-invalid") }}

                        <div class="invalid-feedback">
                            {% for error in form.email.errors %}
                                <span> {{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.email(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
                <div class="form-group">
                    {{ form.password.label(class="form-control-label") }}
                    {% if form.password.errors %}
                        {{ form.password(class="form-control form-control-lg is-invalid") }}

                        <div class="invalid-feedback">
                            {% for error in form.password.errors %}
                                <span> {{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.password(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
                <div class="form-check">
                    {{ form.remember(class="form-check-input") }}
                    {{ form.remember.label(class="form-check-label") }}
                </div>
            </fieldset>
            <div class="form-group">
                {{ form.submit(class="btn btn-outline-info") }}
                <small class="text-muted ml-2">
                    <a href="{{ url_for('users.reset_request') }}">Forgot your
password?</a>
                </small>
            </div>
        </form>
    </div>
    <div class="border-top pt-3">

```



```

        <small class="text-muted">
            Do you need an account? <a class="ml-2" href="{{ url_for('users.reg-
ister') }}">Sign up now</a>
        </small>
    </div>
{% endblock content %}

```

Файл post.html отвечает за отображение поста со всей информацией о блюде: названии, происхождении, категории, ингредиентах, энергетической ценности и подробном описании.

```

{% extends "layout.html" %}
{% block content %}
    <article class="media content-section">
        
        <div class="media-body">
            <div class="article-metadata">
                <a class="mr-2" href="{{ url_for('users.user_posts',
username=post.author.username) }}">{{ post.author.username }}</a>
                <small class="text-muted">{{ post.date_posted.strftime('%d-%m-%Y
%H:%M') }}</small>
            </div>
            {% if post.author == current_user or current_user.role ==
'admin' %}
                <div>
                    <a class="btn btn-secondary btn-sm mt-1 mb-1"
href="{{ url_for('posts.update_post', post_id=post.id) }}">Edit</a>
                    <button type="button" class="btn btn-danger btn-sm
m-1" data-bs-toggle="modal" data-bs-target="#deleteModal">Delete</button>
                </div>
            {% endif %}
            {{ post.like|length }}
            {% if current_user in post.like|map(attribute="author")|list
%}
                <a href="/like-post/{{ post.id }}"><i class="fa-solid
fa-thumbs-up"></i></a>
            {% else %}
                <a href="/like-post/{{ post.id }}"><i class="fa-regular
fa-thumbs-up"></i></a>
            {% endif %}
        </div>
    </div>
    <h1 class="article-title">{{ post.title }}</h1>
    <p class="lead">Origins:</p>
    {% for origin in cuisines %}
        {% for cuisine in post.cuisines %}
            {% if origin.id == cuisine.id %}
                <p>{{ origin.name }}</p>
            {% endif %}
        {% endfor %}
    {% endfor %}
    <p class="lead">Categories:</p>
    {% for category in categories %}
        {% for diet in post.categories %}
            {% if category.id == diet.id %}
                <p>{{ category.name }}</p>
            {% endif %}
        {% endfor %}
    {% endfor %}

```

```

        <p class="lead">Ingredients:</p>
        {% for ingredient in post.ingredients %}
            {% for product in amount %}
                {% if ingredient.id == product[1] %}
                    <p>{{ ingredient.name }}: {{ product[2] }} </p>
                {% endif %}
            {% endfor %}
        {% endfor %}
        <p class="lead">Macronutrients: </p>
        <p>Calories: {{ post.calories }}</p>
        <p>Proteins: {{ post.proteins }}</p>
        <p>Lipids: {{ post.lipids }}</p>
        <p>Carbohydrates: {{ post.carbohydrates }}</p>
        <hr class="hr"/>
        <p style="font-size:20px;" class="article-content">{{ post.content
    }}</p>

</div>
</article>
<div class="content-section ms-5">
    <div class="card-body">
        <div class="card-text"></div>
        <h3>Leave a comment!</h3>
        <br/>
        <div class="collapse" id="comments-{{ post.id }}">
            <div class="card">
                <div class="card-body" id="comments-expanded-{{ post.id }}">
                    {% for comment in post.comments %}
                        <div class="content-section">
                            <div class="f-flex justify-content-between align-
items-center">
                                <a href="{{ url_for('users.user_posts',
username=comment.author.username) }}">{{ comment.author.username }}</a>: {{ com-
ment.text }}
                                <small class="text-muted d-flex justify-content-
end">{{ comment.timestamp.strftime('%d-%m-%Y %H:%M') }}</small>
                                {% if current_user == comment.author or cur-
rent_user == post.author or current_user.role == 'admin' %}
                                    <div class="btn-group">
                                        <button type="button" class="btn btn-sm
btn-primary dropdown-toggle" data-bs-toggle="dropdown"></button>
                                        <ul class="dropdown-menu">
                                            <li>
                                                <a href="/delete-com-
ment/{{comment.id}}" class="dropdown-item">Delete</a>
                                            </li>
                                        </ul>
                                    </div>
                                {% endif %}
                            </div>
                        </div>
                    {% endfor %}
                </div>
            </div>
        </div>
    </div>
    <div class="card-text">
        {% if post.comments|length > 0 %}
            <a data-bs-toggle="collapse" href="#comments-{{ post.id }}"
role="button">
                <small>View/Hide {{ post.comments|length }} Com-
ments</small>
            </a>
        {% else %}

```

```

        <small class="text-muted">Become the first one to re-
view!</small>
        {% endif %}
    </p>
    <form class="input-group" method="POST" action="/create-comment/{{
post.id }}">
        <input type="text" id="text" name="text" class="form-control"
placeholder="Have something to say?"/>
        <button type="submit" class="btn btn-outline-primary">Sub-
mit</button>
    </form>
</div>
</div>
<!-- Modal -->
<div class="modal fade" id="deleteModal" tabindex="-1" aria-la-
belledby="deleteModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="deleteModalLabel">Delete
post?</h5>
                <button type="button" class="btn-close" data-bs-dis-
miss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">
                <p>You won't be able to cancel this action.</p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-dis-
miss="modal">Close</button>
                <form action="{{ url_for('posts.delete_post',
post_id=post.id) }}" method="POST">
                    <input class="btn btn-danger" type="SUBMIT" value="De-
lete">
                </form>
            </div>
        </div>
    </div>
</div>
</div>
{% endblock content %}

```

Файл register.html – страница регистрации на сайте.

```

{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Join Today</legend>
                <div class="form-group">
                    {{ form.username.label(class="form-control-label") }}

                    {% if form.username.errors %}
                        {{ form.username(class="form-control form-control-lg is-
invalid") }}

                        <div class="invalid-feedback">
                            {% for error in form.username.errors %}
                                <span> {{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.username(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
            </fieldset>
        </form>
    </div>
{% endblock %}

```

```

}}
        {% endif %}
    </div>

    <div class="form-group">
        {{ form.email.label(class="form-control-label") }}

        {% if form.email.errors %}
            {{ form.email(class="form-control form-control-lg is-in-
valid") }}

            <div class="invalid-feedback">
                {% for error in form.email.errors %}
                    <span> {{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ form.email(class="form-control form-control-lg") }}
        {% endif %}
    </div>

    <div class="form-group">
        {{ form.password.label(class="form-control-label") }}

        {% if form.password.errors %}
            {{ form.password(class="form-control form-control-lg is-
invalid") }}

            <div class="invalid-feedback">
                {% for error in form.password.errors %}
                    <span> {{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ form.password(class="form-control form-control-lg")
}}

        {% endif %}
    </div>

    <div class="form-group">
        {{ form.confirm_password.label(class="form-control-label")
}}

        {% if form.confirm_password.errors %}
            {{ form.confirm_password(class="form-control form-con-
trol-lg is-invalid") }}

            <div class="invalid-feedback">
                {% for error in form.confirm_password.errors %}
                    <span> {{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ form.confirm_password(class="form-control form-con-
trol-lg") }}

        {% endif %}
    </div>
</fieldset>
<div class="form-group">
    {{ form.submit(class="btn btn-outline-info") }}
</div>
</form>
</div>
<div class="border-top pt-3">
    <small class="text-muted">
        Already have an account? <a class="ml-2" href="{{ url_for('us-
ers.login') }}">Sign in</a>

```

```

        </small>
    </div>
{% endblock content %}

```

Файл `reset_request.html` загружает страницу для отправки сообщения с ссылкой на форму сброса пароля.

```

{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Reset Password</legend>
                <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}

                    {% if form.email.errors %}
                        {{ form.email(class="form-control form-control-lg is-in-
valid") }}

                        <div class="invalid-feedback">
                            {% for error in form.email.errors %}
                                <span> {{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.email(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
            </fieldset>
            <div class="form-group">
                {{ form.submit(class="btn btn-outline-info") }}
            </div>
        </form>
    </div>
{% endblock content %}

```

Файл `reset_token.html` – непосредственно форма сброса пароля.

```

{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Reset Password</legend>
                <div class="form-group">
                    {{ form.password.label(class="form-control-label") }}

                    {% if form.password.errors %}
                        {{ form.password(class="form-control form-control-lg is-
invalid") }}

                        <div class="invalid-feedback">
                            {% for error in form.password.errors %}
                                <span> {{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.password(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
            </fieldset>
        </form>
    </div>

```

```

        <div class="form-group">
            {{ form.confirm_password.label(class="form-control-label")
        }}

        {% if form.confirm_password.errors %}
            {{ form.confirm_password(class="form-control form-con-
trol-lg is-invalid") }}
            <div class="invalid-feedback">
                {% for error in form.confirm_password.errors %}
                    <span> {{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ form.confirm_password(class="form-control form-con-
trol-lg") }}
        {% endif %}
    </div>
</fieldset>
<div class="form-group">
    {{ form.submit(class="btn btn-outline-info") }}
</div>
</form>
</div>
{% endblock content %}

```

Файл `searched.html` загружает страницу с результатами поиска по ключевым словам среди имеющихся постов.

```

{% extends "layout.html" %}
{% block content %}
    <br/>
    <h2>Your search result for...</h2>
    <h4>'{{ searched }}'</h4>
    <br/>

    {% for post in posts %}
        <article class="media content-section">
            
            <div class="media-body">
                <div class="article-metadata">
                    <a class="mr-2" href="{{ url_for('users.user_posts',
username=post.author.username) }}">{{ post.author.username }}</a>
                    <small class="text-muted">{{ post.date_posted.strftime('%d-
%m-%Y %H:%M') }}</small>
                </div>
                <h2><a class="article-title" href="{{ url_for('posts.post',
post_id=post.id) }}">{{ post.title }}</a></h2>
                <p class="article-content">{{ post.content }}</p>
            </div>
        </article>
    {% endfor %}
{% endblock content %}

```

Файл `users_posts.html` выводит список постов авторства одного и того же пользователя.

```

{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        
    <div class="media-body">
        <h2 class="account-heading">{{ user.username }}</h2>
        <p class="text-secondary">E-mail: {{ user.email }}</p>
        <p class="text-secondary">On website since: {{
user.reg_date.strftime('%d-%m-%Y') }}</p>
        {% if not(user.bio) %}
            <p class="text-secondary">Bio: Not set yet</p>
        {% else %}
            <p class="text-secondary">Bio: "{{ user.bio }}"</p>
        {% endif %}
    </div>
</div>
<h1 class="mb-3">Posts by {{ user.username }} ({{ posts.total }})</h1>
{% for post in posts.items %}
    <article class="media content-section">
        
        <div class="media-body">
            <div class="article-metadata">
                <a class="mr-2" href="{{ url_for('users.user_posts',
username=post.author.username) }}">{{ post.author.username }}</a>
                <small class="text-muted">{{ post.date_posted.strftime('%d-
%m-%Y %H:%M') }}</small>
            </div>
            <h2><a class="article-title" href="{{ url_for('posts.post',
post_id=post.id) }}">{{ post.title }}</a></h2>
            <p class="article-content">{{ post.content }}</p>
        </div>
    </article>
{% endfor %}
{% for page_num in posts.iter_pages(left_edge=1, right_edge=1, left_cur-
rent=1, right_current=1) %}
    {% if page_num %}
        {% if posts.page == page_num %}
            <a class="btn btn-info mb-4" href="{{ url_for('us-
ers.user_posts', username=user.username, page=page_num) }}">{{ page_num }}</a>
        {% else %}
            <a class="btn btn-outline-info mb-4" href="{{ url_for('us-
ers.user_posts', username=user.username, page=page_num) }}">{{ page_num }}</a>
        {% endif %}
    {% else %}
        ...
    {% endif %}
{% endfor %}
{% endblock content %}

```

Файлы 403.html, 404.html, 405.html и 500.html – страницы персонализированных ошибок, вызываемые при ошибках с соответствующими номерами.

Файл 403.html:

```

{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <h1>Not enough rights (403)</h1>
        <p>Oops! Seems, you are not allowed to be here :0 Please check your ac-
count and try again!</p>
    </div>
{% endblock content %}

```

Файл 404.html:

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <h1>Oops. Page not found (404)</h1>
        <p>This page does not exist :c Maybe it will be created one day... But
for now, please, try a different location ;)</p>
    </div>
{% endblock content %}
```

Файл 405.html:

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <h1>Forbidden method! (405)</h1>
        <p>Oops, something went wrong while processing the method for current
URL, you are not supposed to see this message! >c</p>
    </div>
{% endblock content %}
```

Файл 500.html:

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <h1>Something went wrong (500)</h1>
        <p>The server currently feels under the weather. We are making our best
to fix it! Please come back later :3</p>
    </div>
{% endblock content %}
```

ОПИСАНИЕ СЦЕНАРИЕВ ИСПОЛЬЗОВАНИЯ СО СНИМКАМИ ЭКРАНА

При входе на сайт пользователь видит главный экран приложения, где (при наличии таковых) отображаются все уже созданные посты другими пользователями.

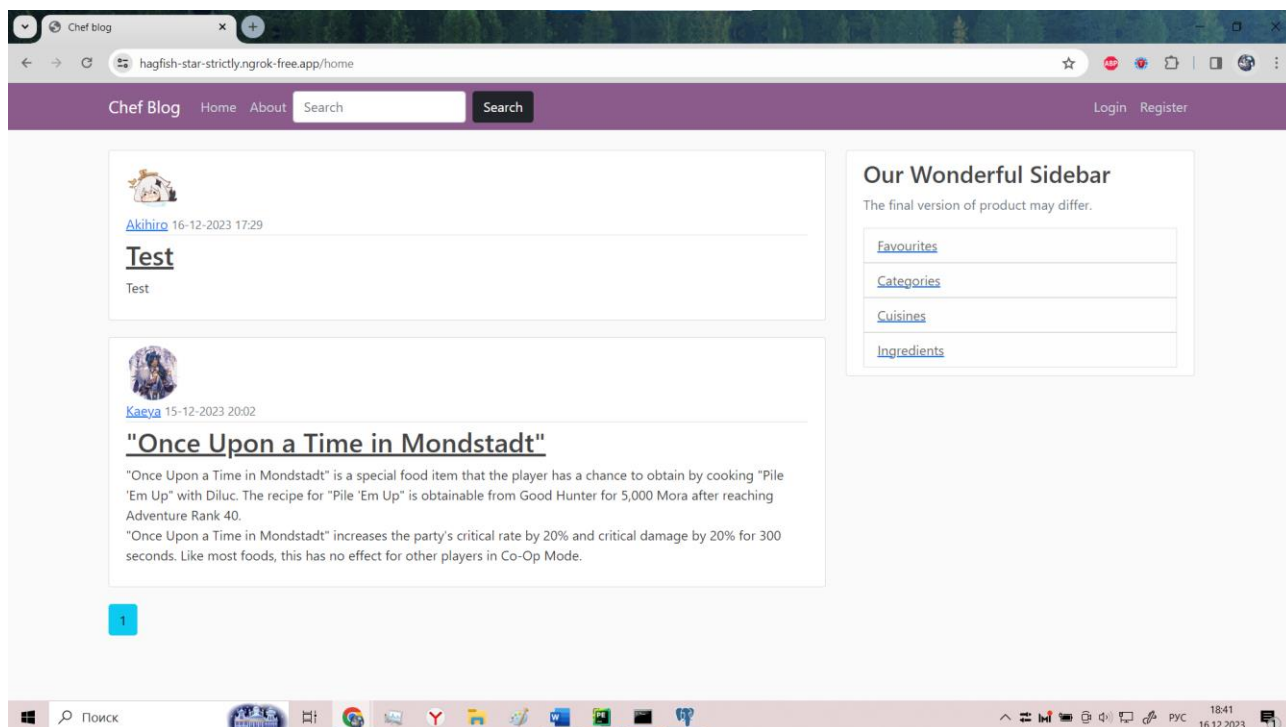


Рисунок 4 – Главный экран

Далее, пользователь может или создать новый аккаунт при нажатии на кнопку Register, тогда его перенаправит на страницу регистрации (рисунок 5), или войти в уже имеющийся аккаунт (рисунок 6). Со страницы входа можно перейти на страницу создания аккаунта (кнопка «Sign in») и наоборот (кнопка «Sign up now»).

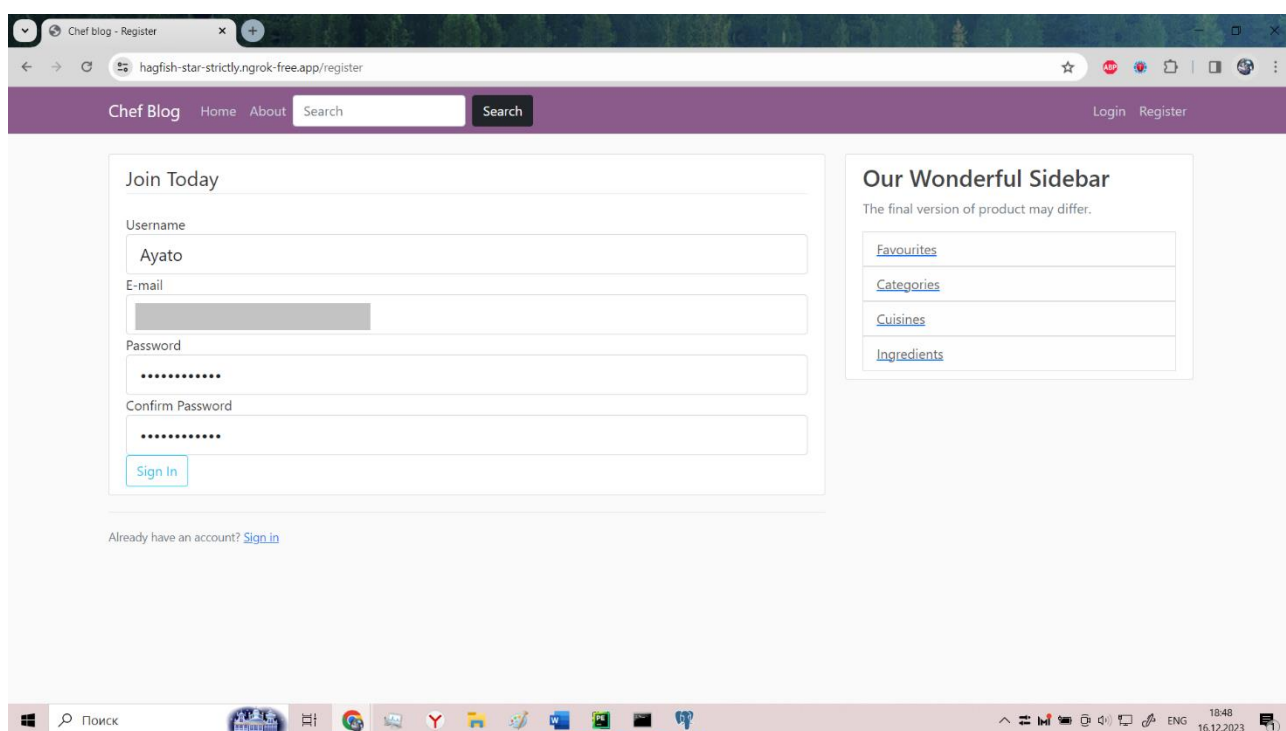


Рисунок 5 – Страница регистрации

После регистрации пользователя оповещают об успешном создании аккаунта и перенаправляют на страницу входа в учетную запись.

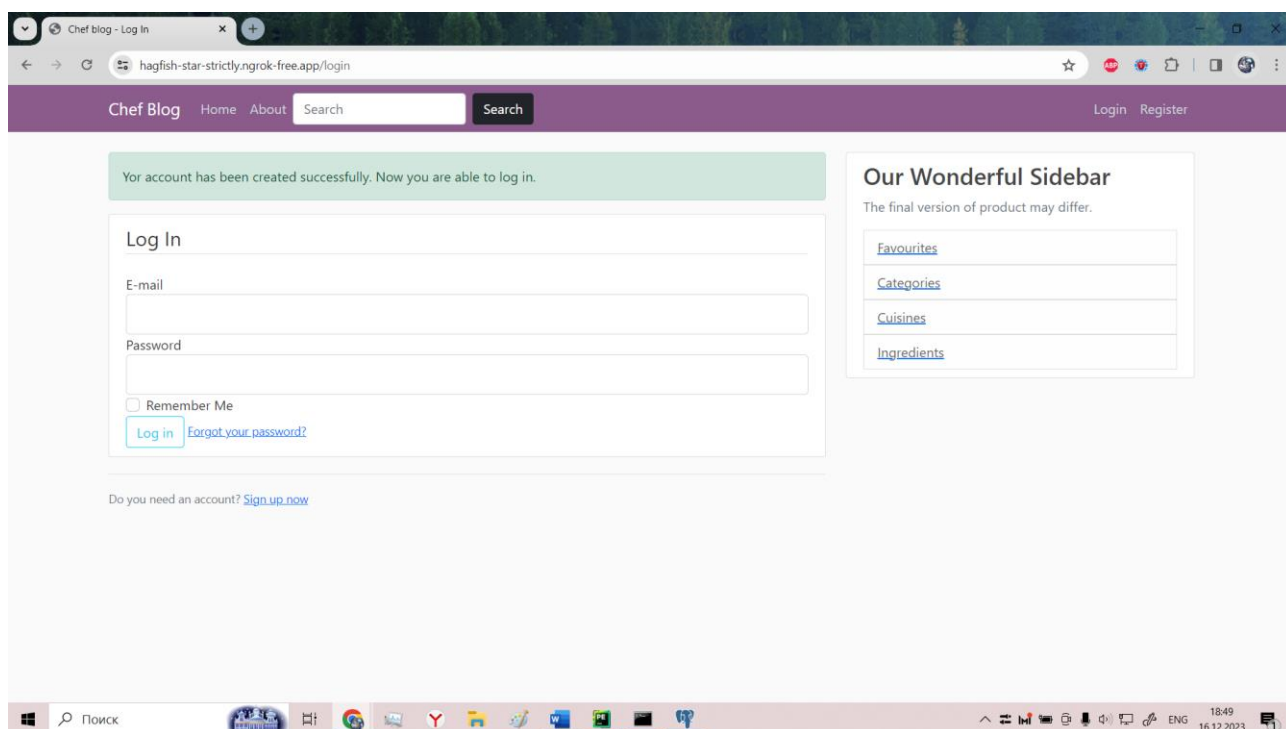


Рисунок 6 – Страница входа в приложение

Если гость забыл свой пароль, то по кнопке «Forgot your password?» его можно сбросить. Пользователя попросят ввести адрес электронной почты, на которую был зарегистрирован аккаунт, и отправят на нее письмо со ссылкой.

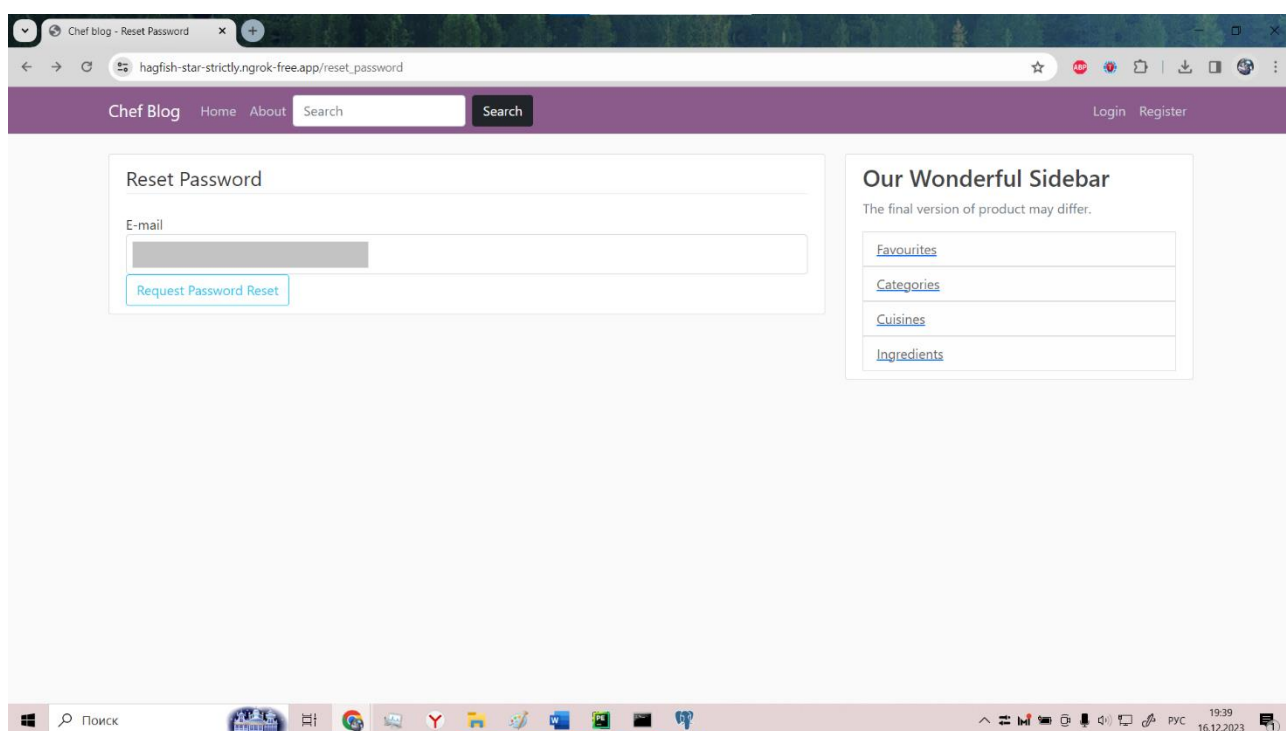


Рисунок 7 – Запрос ссылки на сброс пароля



Рисунок 8 – Письмо с ссылкой

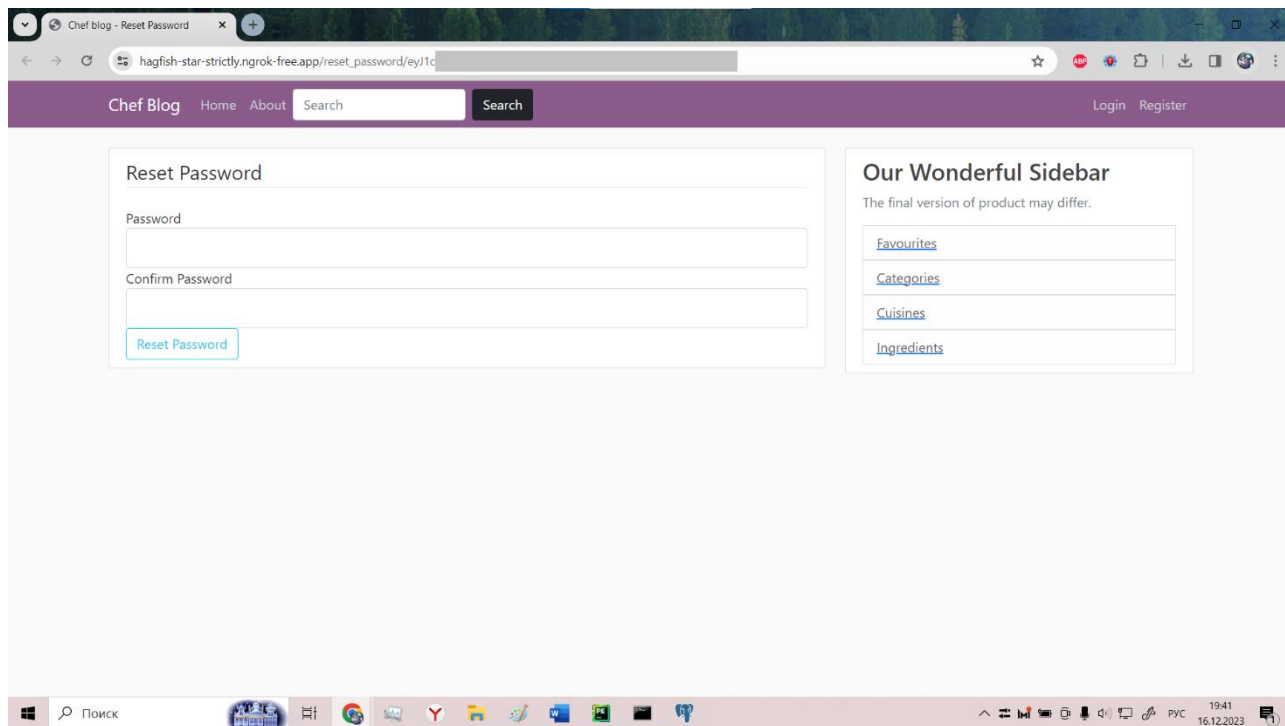


Рисунок 9 – Форма сброса пароля

После успешной смены пароля, пользователь будет оповещен об этом и перенаправлен на страницу входа в учетную запись.

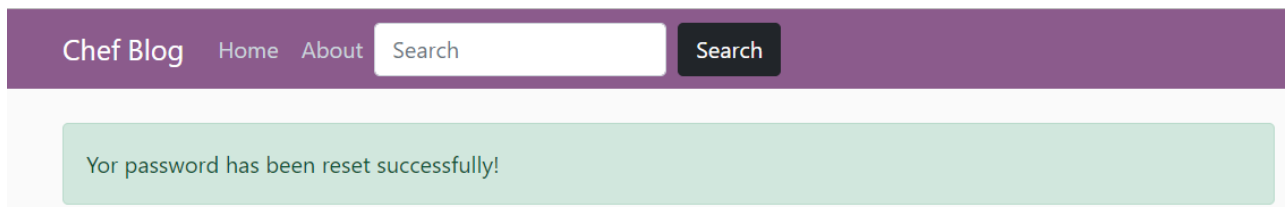


Рисунок 10 – Сообщение об успешной смене пароля

После входа в аккаунт, пользователя перебрасывает на главную страницу и приветствуют. На панели навигации появляется имя текущего пользователя и становятся доступны кнопки создания поста и выхода из аккаунта.

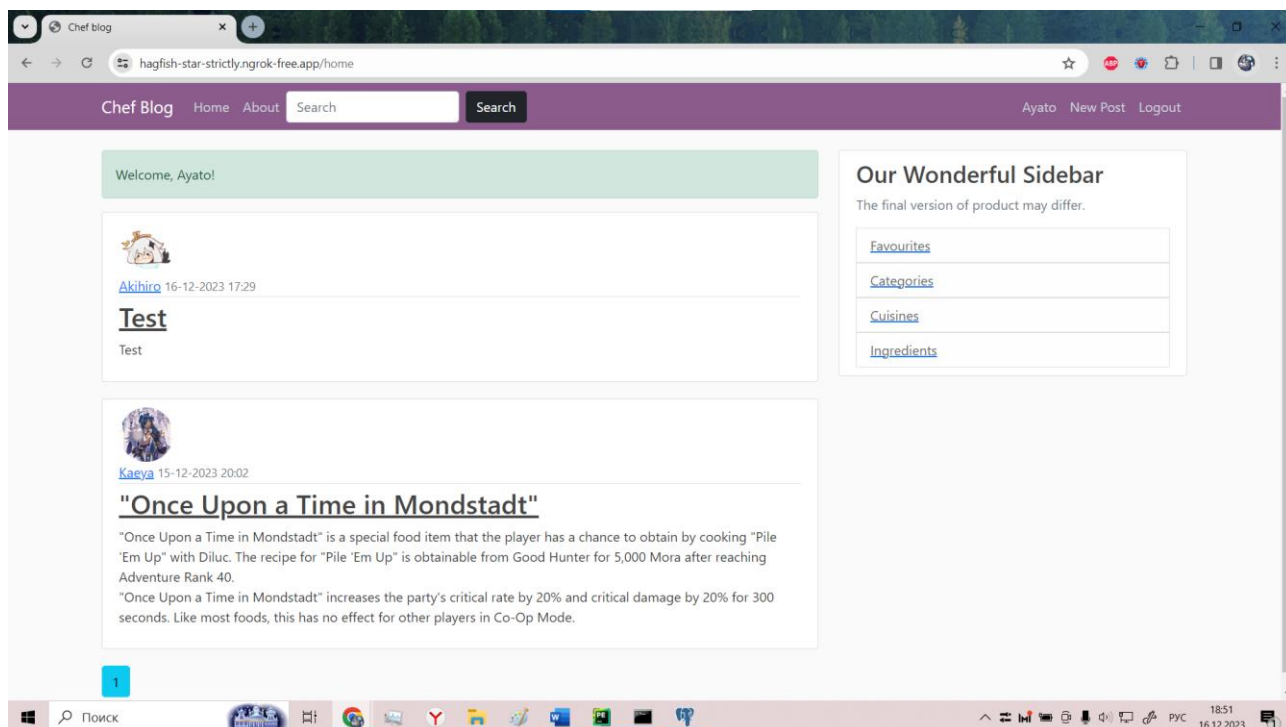


Рисунок 11 – Приветствие нового пользователя

При нажатии на имя пользователя в верхнем правом углу страницы, открывается профиль, где отображаются имя, роль, e-mail, дата регистрации и краткие сведения о себе, с возможностью его отредактировать (обновить фотографию профиля, изменить имя, адрес электронной почты или биографию). По нажатии на кнопку «Update» внесенные изменения сохраняются.

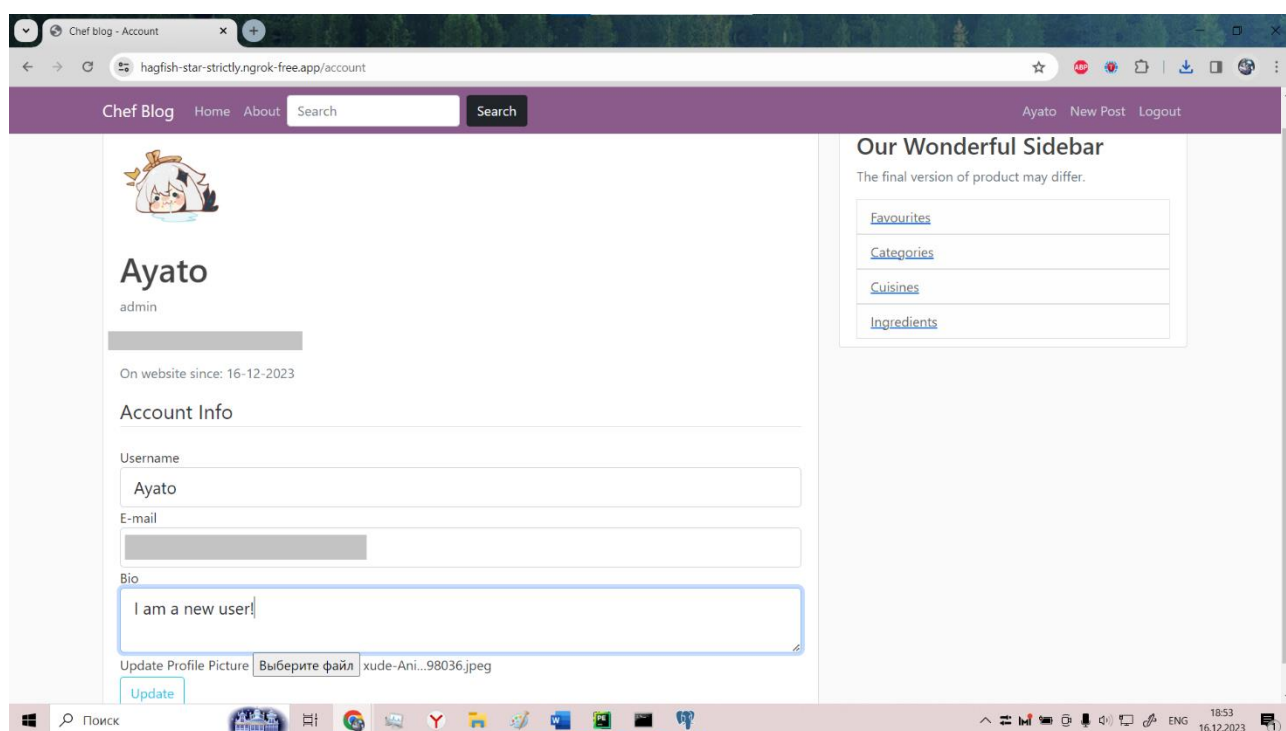


Рисунок 12 – Профиль

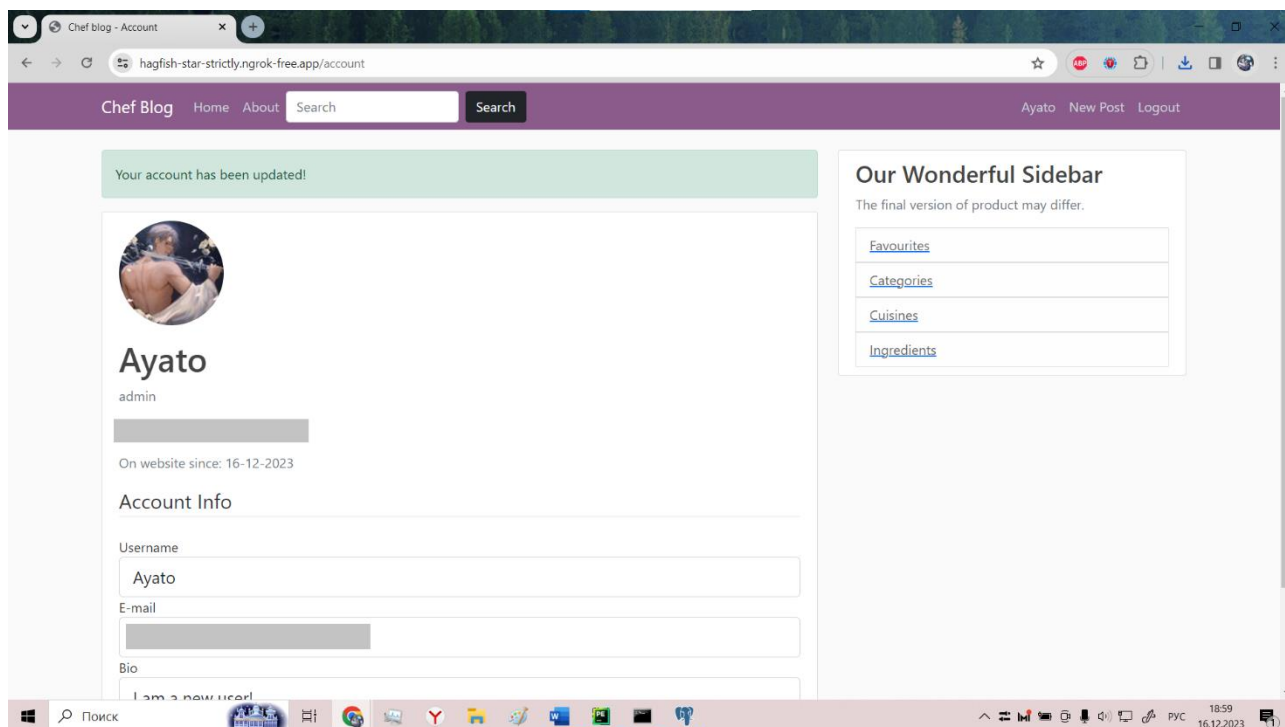


Рисунок 13 – Обновленный профиль и сообщение об успехе

Для создания нового поста необходимо нажать на кнопку New Post, тогда откроется форма для создания рецепта, где помимо описания процесса приготовления нужно указать название блюда, его происхождение и категории из выпадающих списков, а также ингредиенты с указанием количества.

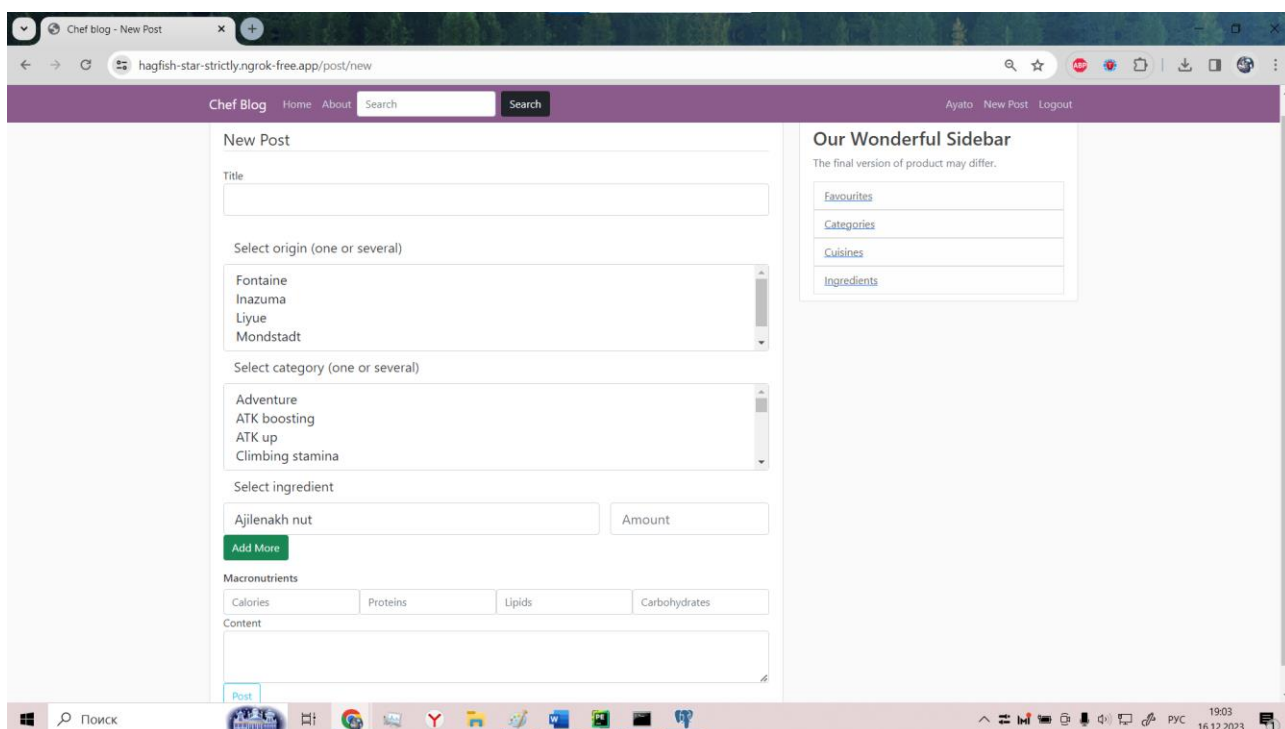


Рисунок 14 – Форма создания рецепта

Для добавления дополнительных строк для указания ингредиентов из выбранных, нужно нажать на кнопку «Add More».

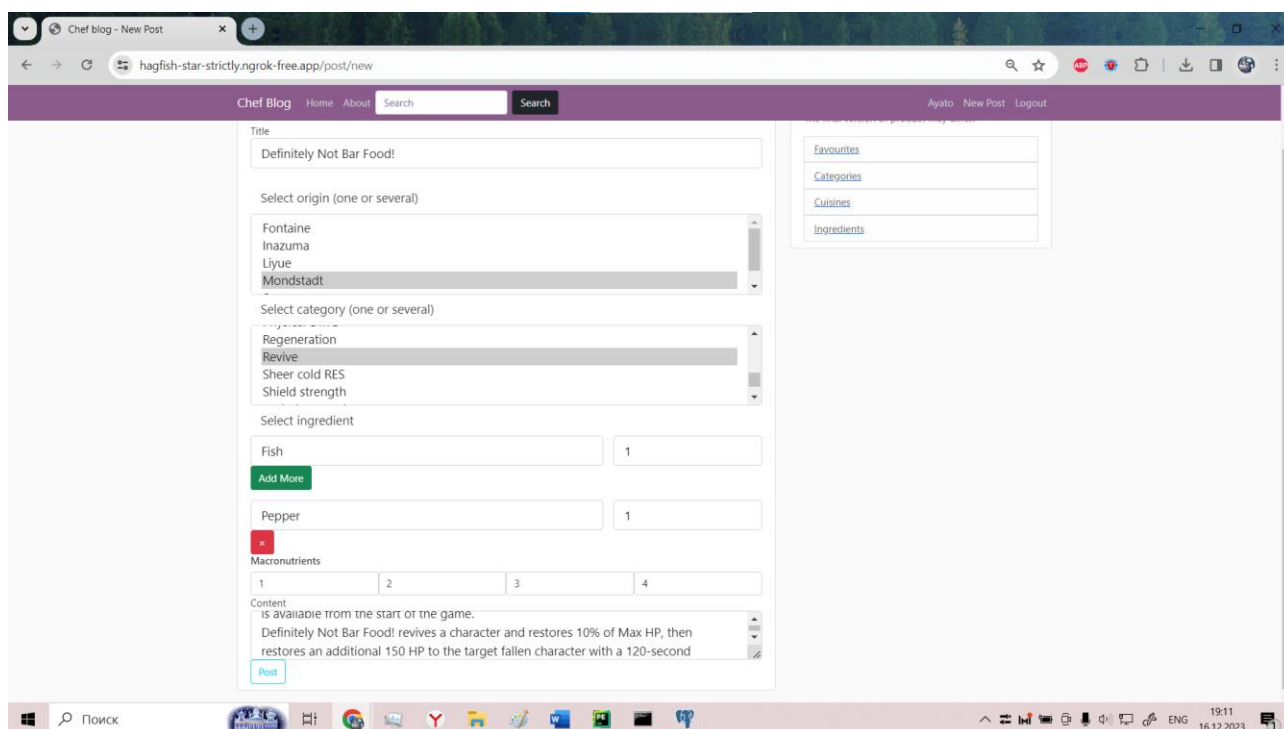


Рисунок 15 – Заполненная форма

При нажатии на кнопку «Post» запись опубликуется и появится на главном экране приложения. Ее можно просмотреть, нажав на название. Откроется страница со всеми указанными при создании данными, числом лайков, комментариями и кнопками для редактирования и удаления.

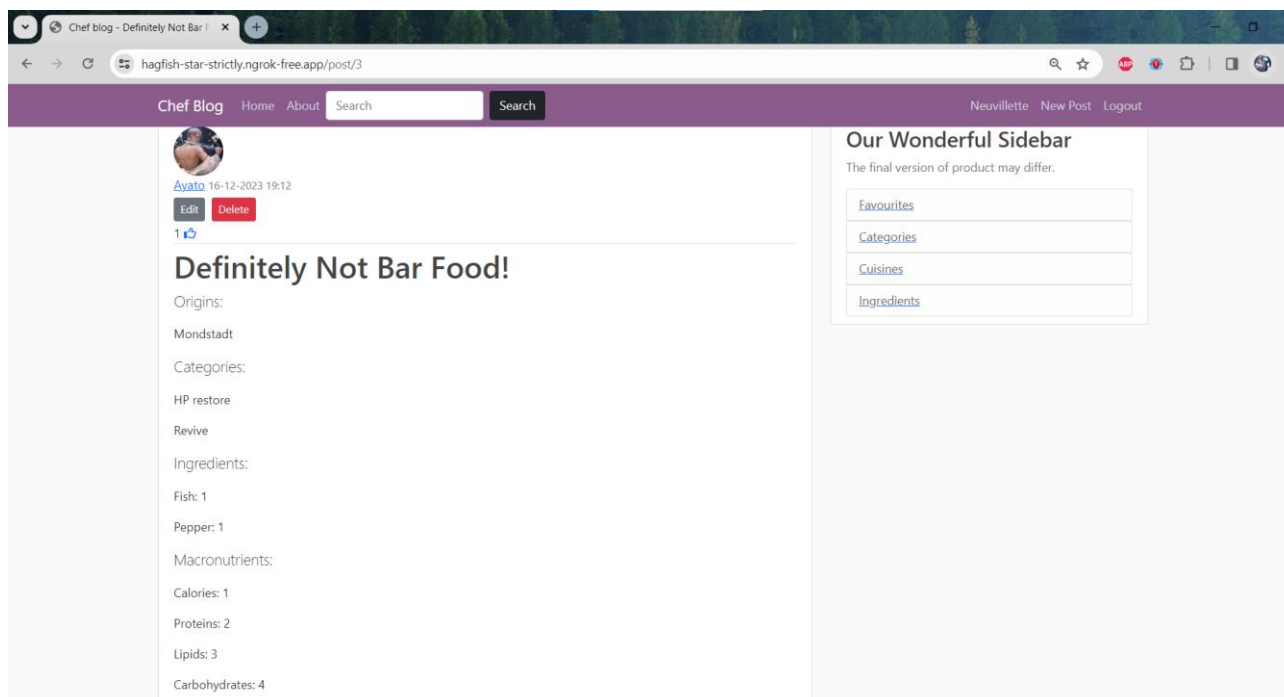


Рисунок 16 – Шапка поста

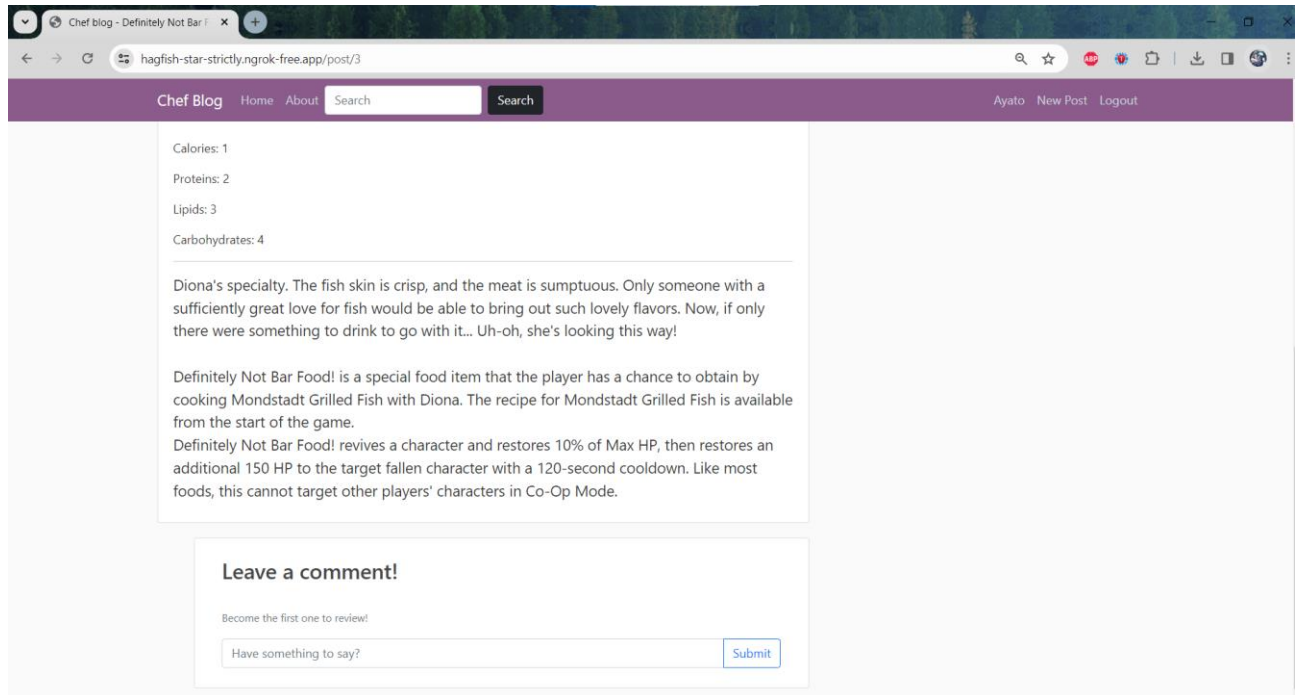


Рисунок 17 – Содержание рецепта и комментарии

При нажатии на кнопку «About» открывается страница с информацией о приложении.

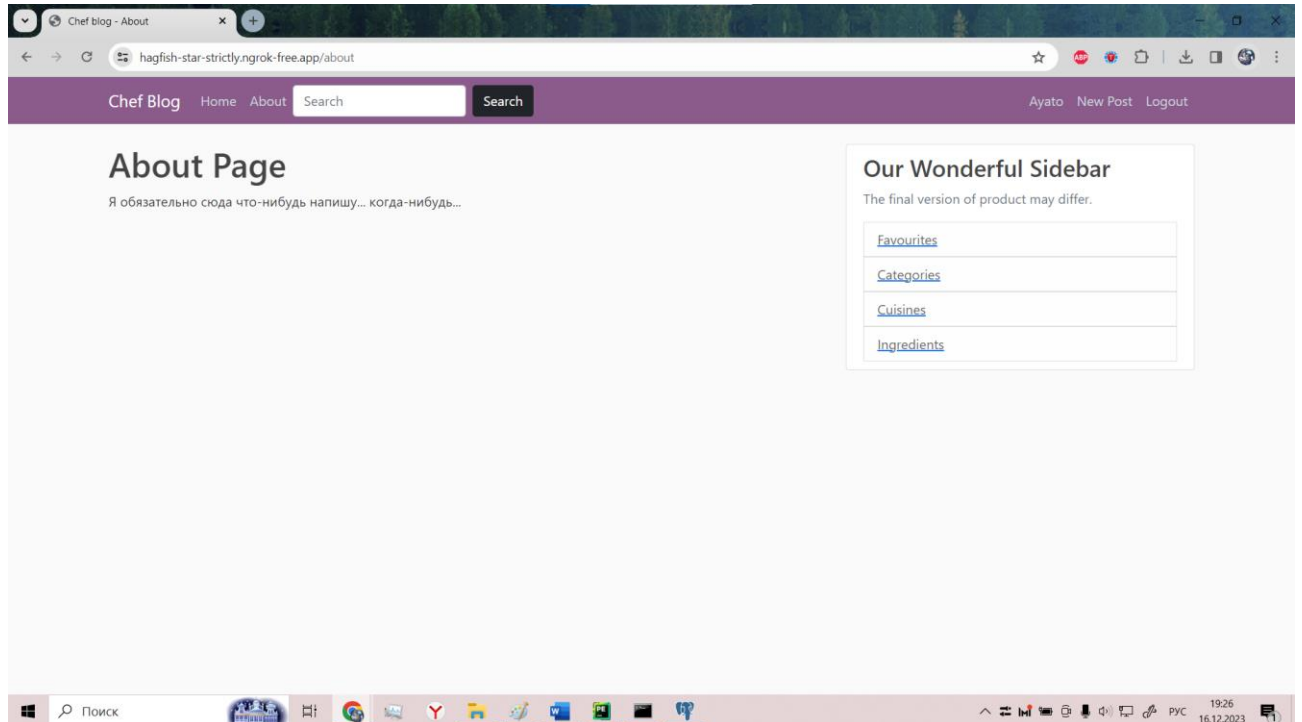


Рисунок 18 – Страница «About»

Пользователю доступны список избранных постов (рецептов, отмеченных лайком), ингредиентов, категорий и кухонь, разбитых на страницы, из бокового меню.

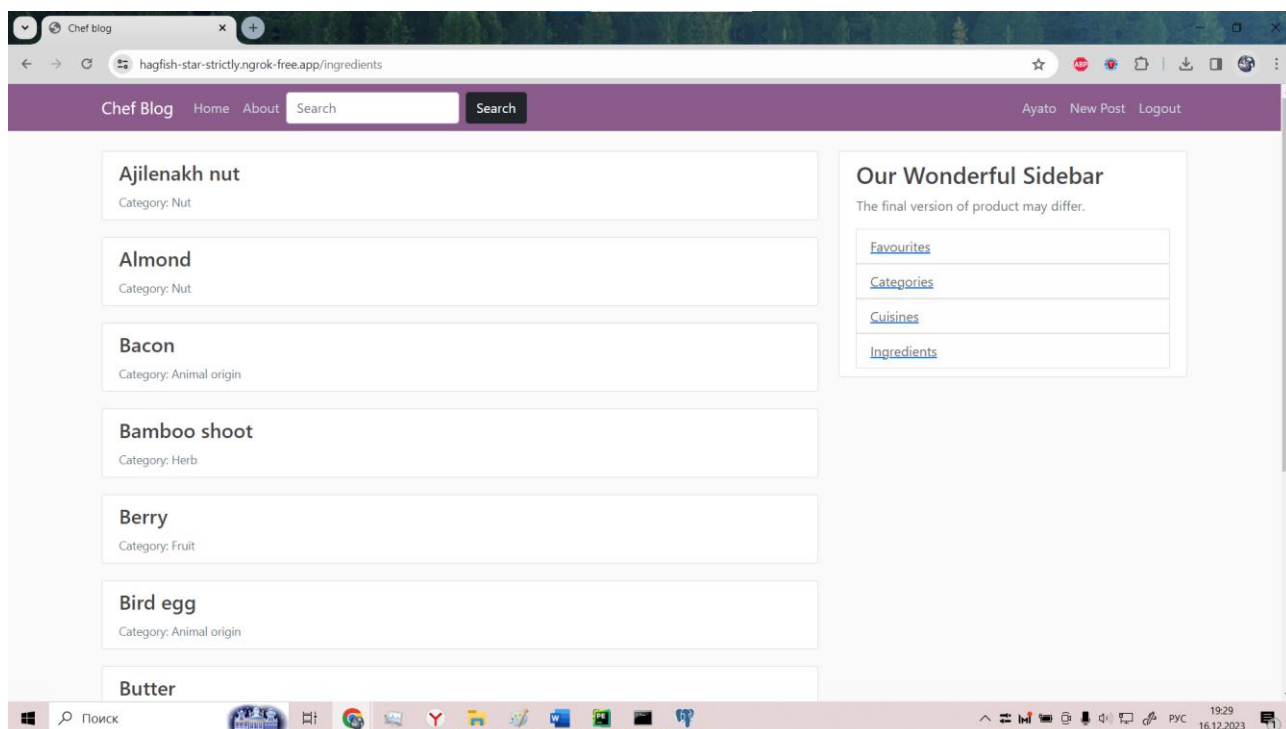


Рисунок 19 – Список ингредиентов и их категория

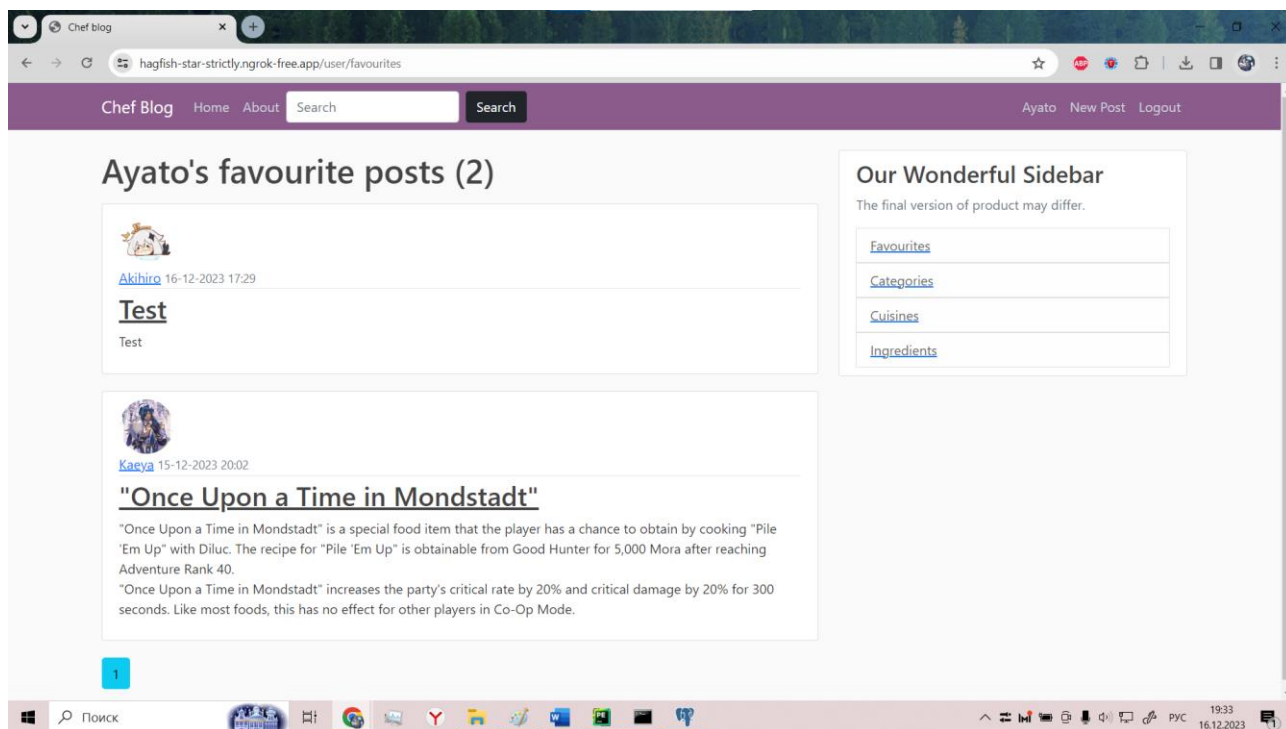


Рисунок 20 – Избранные посты

Все кухни и категории являются фильтрами. При нажатии на какую-либо из ссылок, будет выведен список рецептов, относящихся к выбранной кухне или категории (рисунок 23).

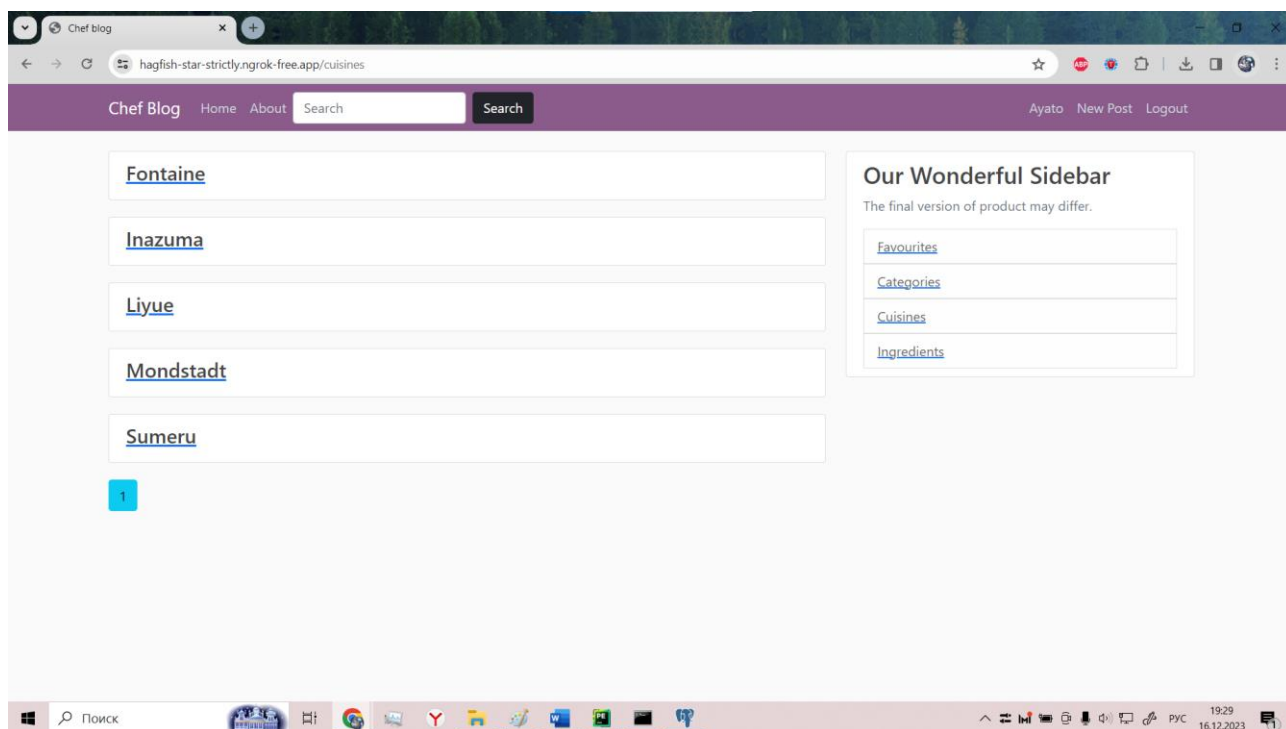


Рисунок 21 – Список кухонь

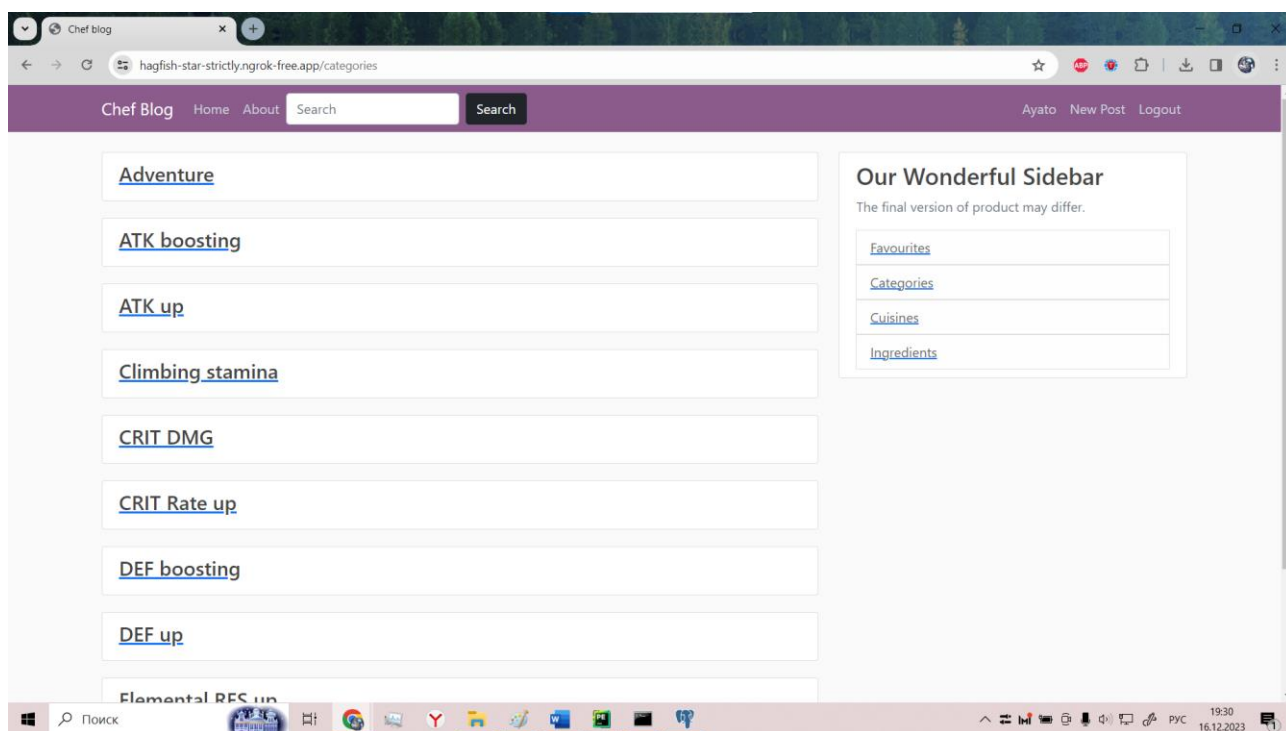


Рисунок 22 – Список категорий

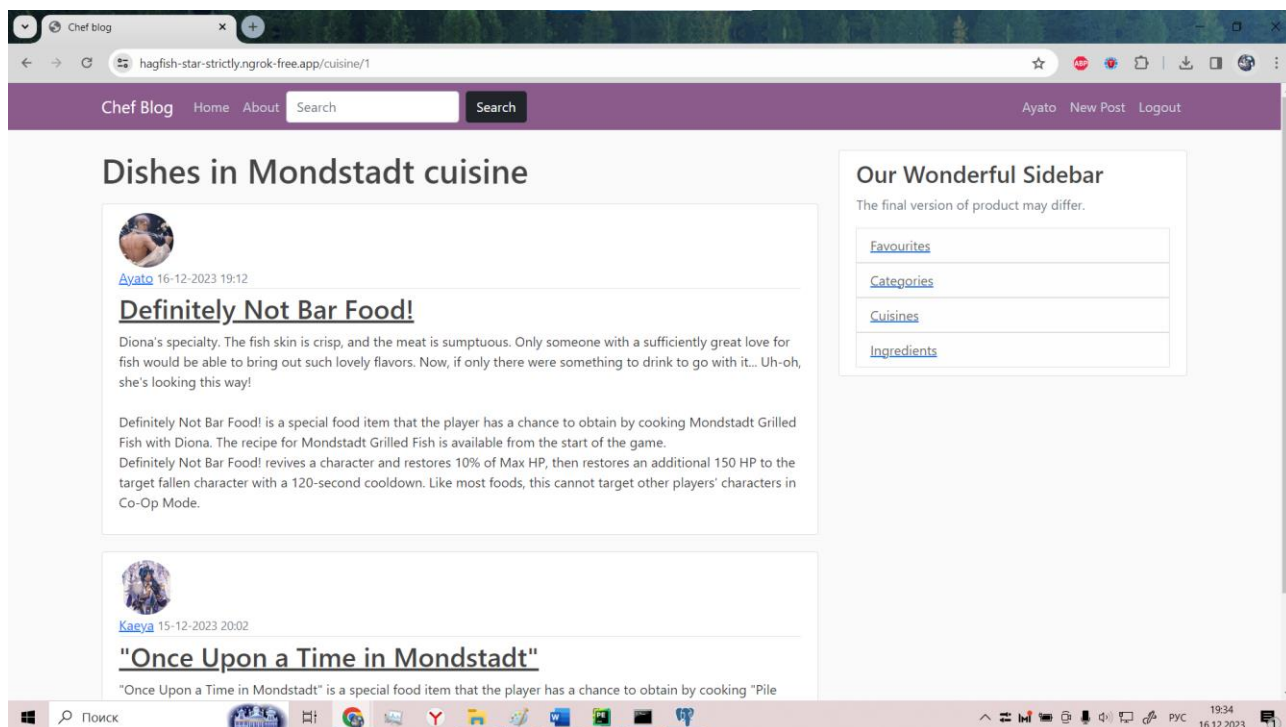


Рисунок 23 – Блюда Мондштадтской кухни

Если пользователь является админом, то ему доступно редактирование удаление чужих постов и комментариев. Простой пользователь может удалять вышеперечисленное только своего авторства. При попытке удалить пост, пользователя попросят подтвердить свои намерения во всплывающем сообщении (рисунок 24).

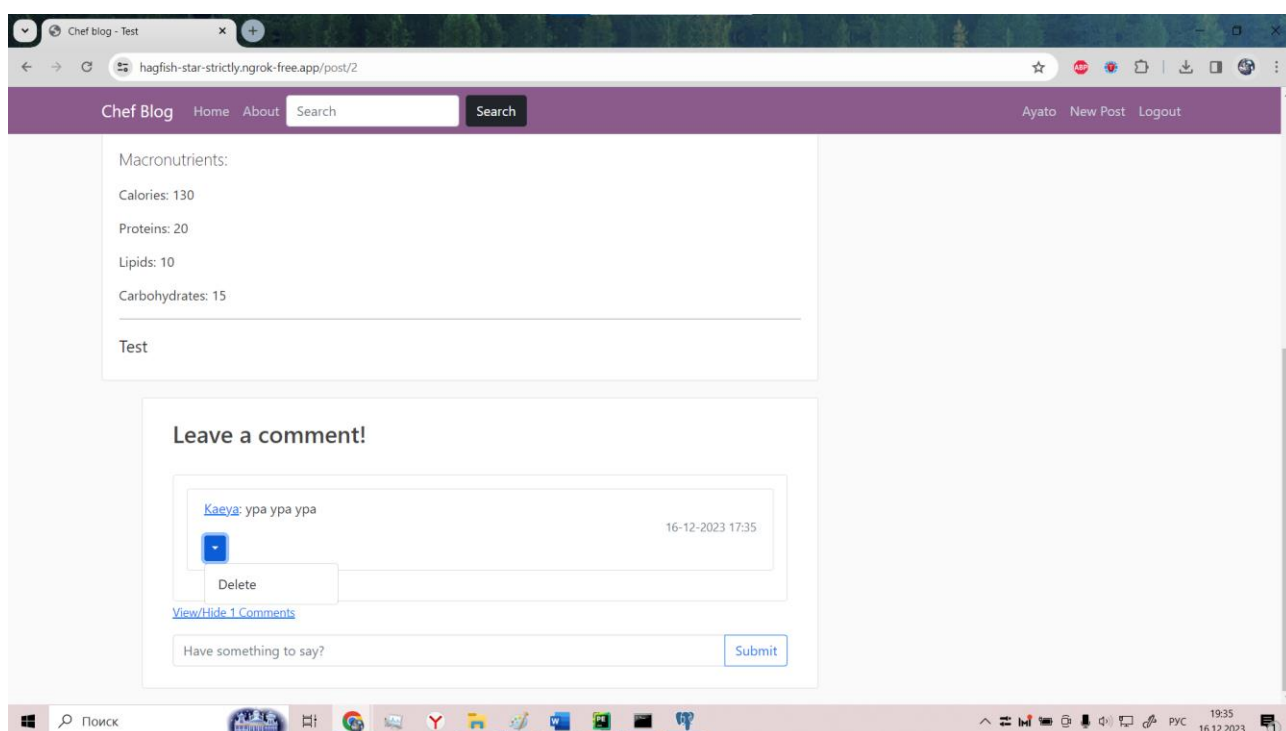


Рисунок 24 – Кнопка удаления комментария

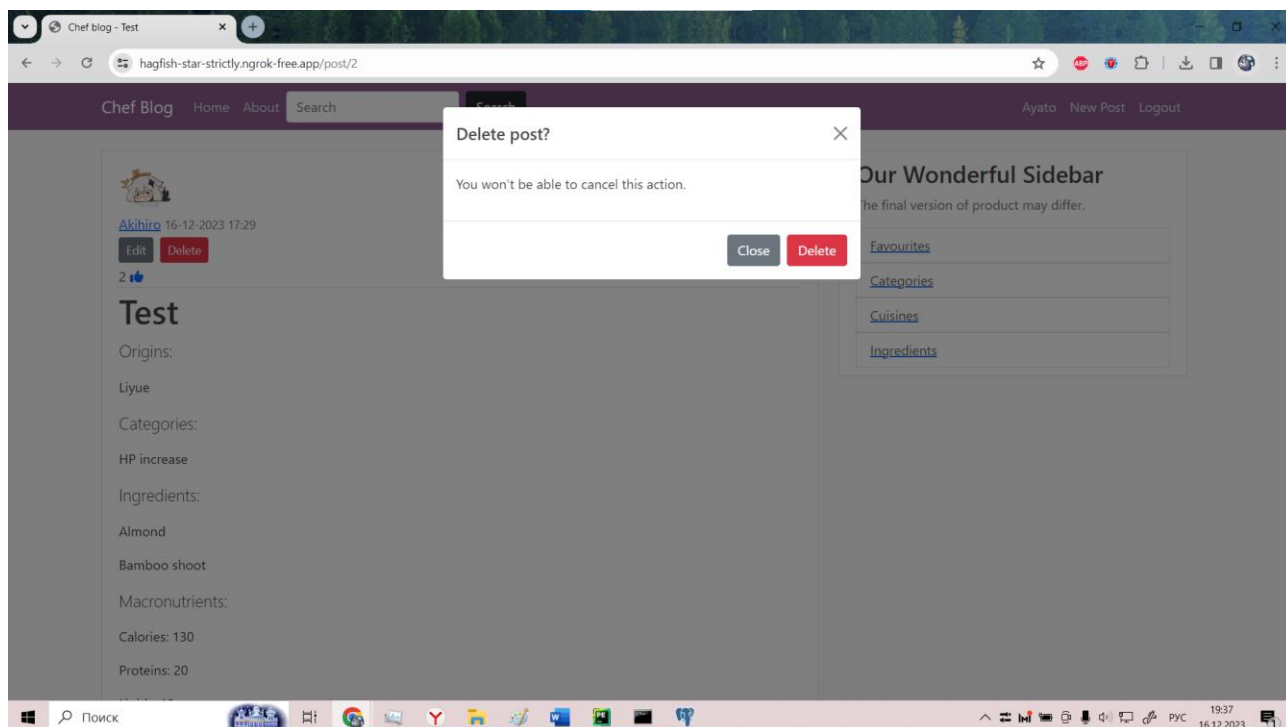


Рисунок 25 – Сообщение с подтверждением действия

При нажатии на имя пользователя в шапке поста, откроется список рецептов его авторства и краткая информация о пользователе.

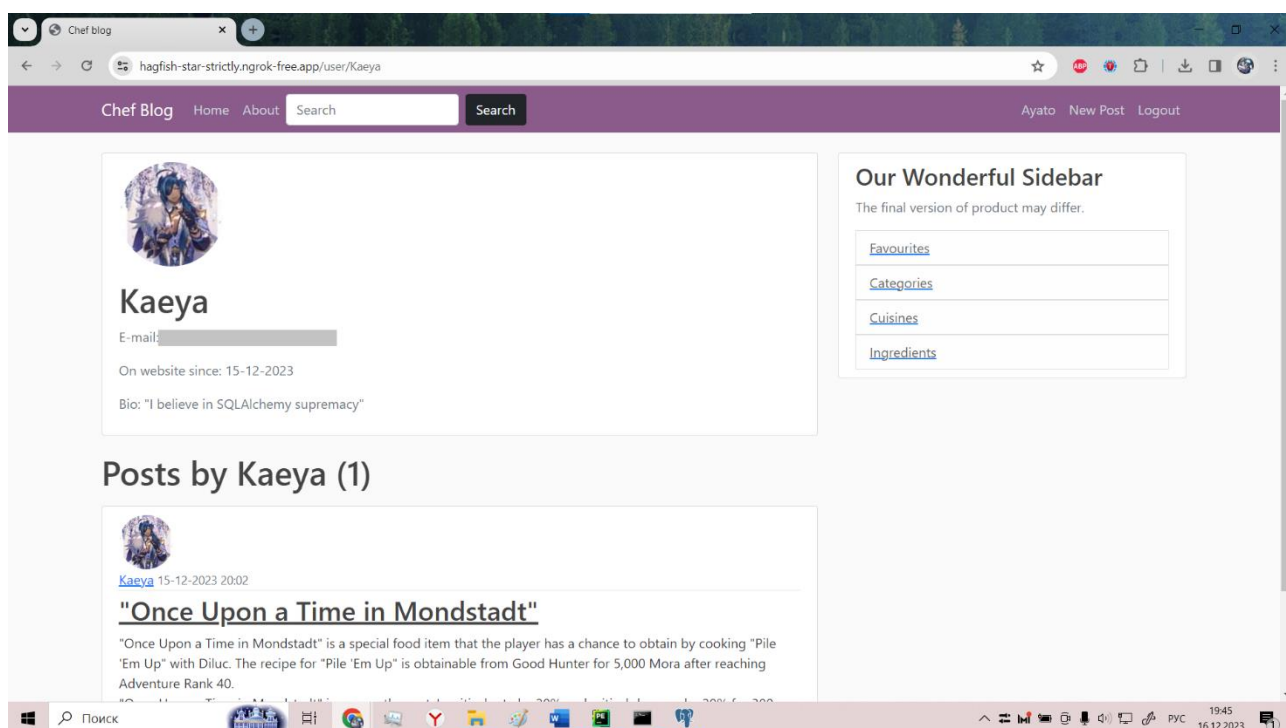


Рисунок 26 – Рецепты конкретного пользователя

Гостям и зарегистрированным пользователям доступен поиск по ключевым словам рецепта в шапке приложения. Результатом будет список постов, содержащих указанное слово или набор слов.

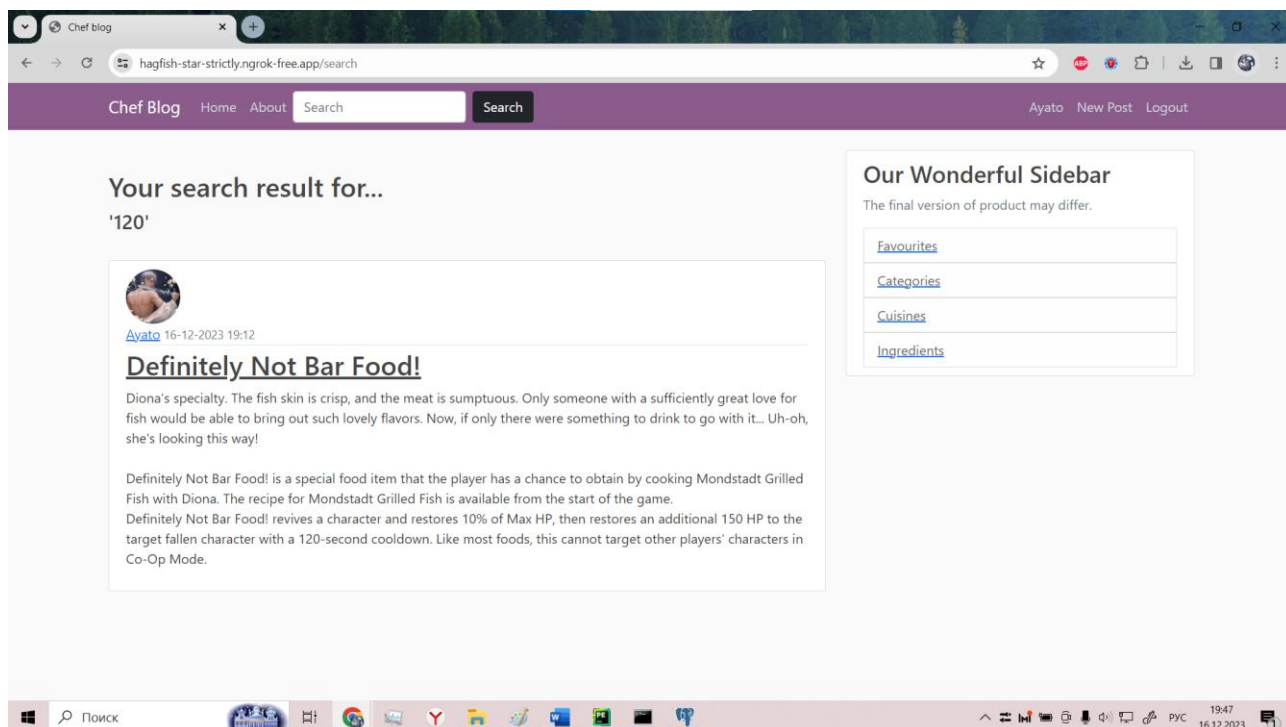


Рисунок 27 – Результат поиска по словам среди рецептов

ЗАКЛЮЧЕНИЕ

В работе описан процесс создания веб-приложения, представляющего из себя кулинарную книгу, на языке программирования Python с использованием фреймворка Flask. В качестве дополнительных модулей использовались Bootstrap-5, FontAwesome и SQLAlchemy.

В ходе выполнения курсовой работы были усовершенствованы навыки ПП, работы с базами данных и написания и оформления курсовых работ. Графическая и программная составляющие приложения выполнены в соответствии с техническим заданием, все элементы приложения отображаются и функционируют корректно.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Stackoverflow (IT-блог): <https://stackoverflow.com/>
2. YouTube URL: [Электронный ресурс] -
<https://www.youtube.com/@coreyms>
3. YouTube URL: [Электронный ресурс] -
<https://www.youtube.com/@TechWithTim>
4. YouTube URL: [Электронный ресурс] -
<https://www.youtube.com/@Codemycom>
5. Devman: <https://dvmn.org/encyclopedia/web-server/ngrok/>
6. Документация SQLAlchemy: <https://www.sqlalchemy.org/>
7. Аксёнов А.В. Разработка и развертывание веб-приложения на языке Python [Электронный ресурс] - <https://github.com/db-course/course-project-manual/blob/master/index.rst>