Факультет компьютерных наук

Образовательная программа
«Программная инженерия"

Москва 2024

КофеСкаут (CoffeeScout)

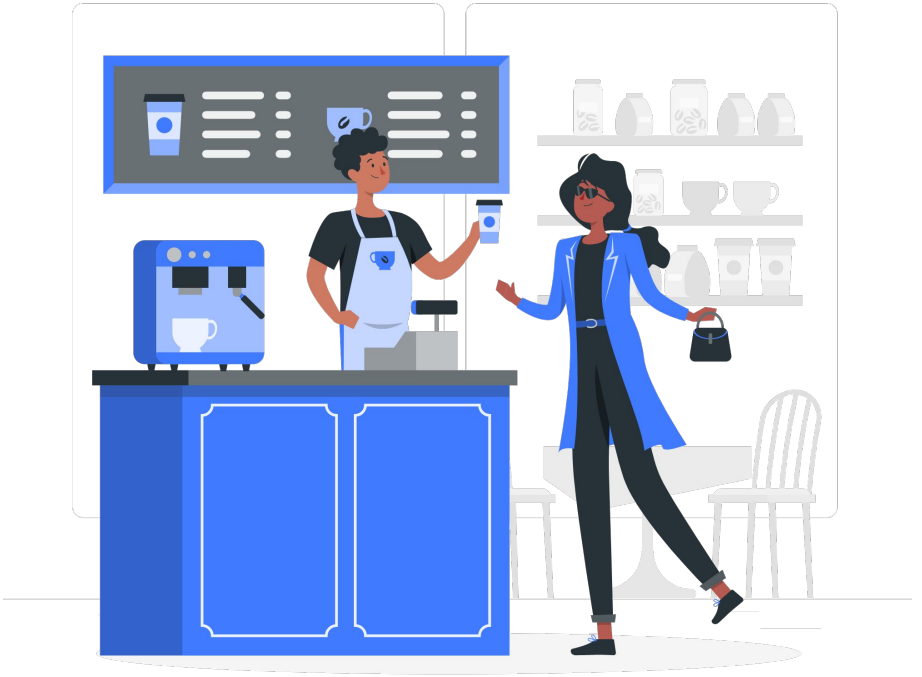
Командный программный проект

Выполнили: студенты БПИ-228 Ифраимова Майя Альбертовна и Иванов Максим Романович
Научный руководитель: Виденин С.А., преподаватель ДПИ ФКН

Описание предметной области

«КофеСкаут» - приложение для поиска интересующего напитка в кофейнях, расположенных в непосредственной близости от пользователя.

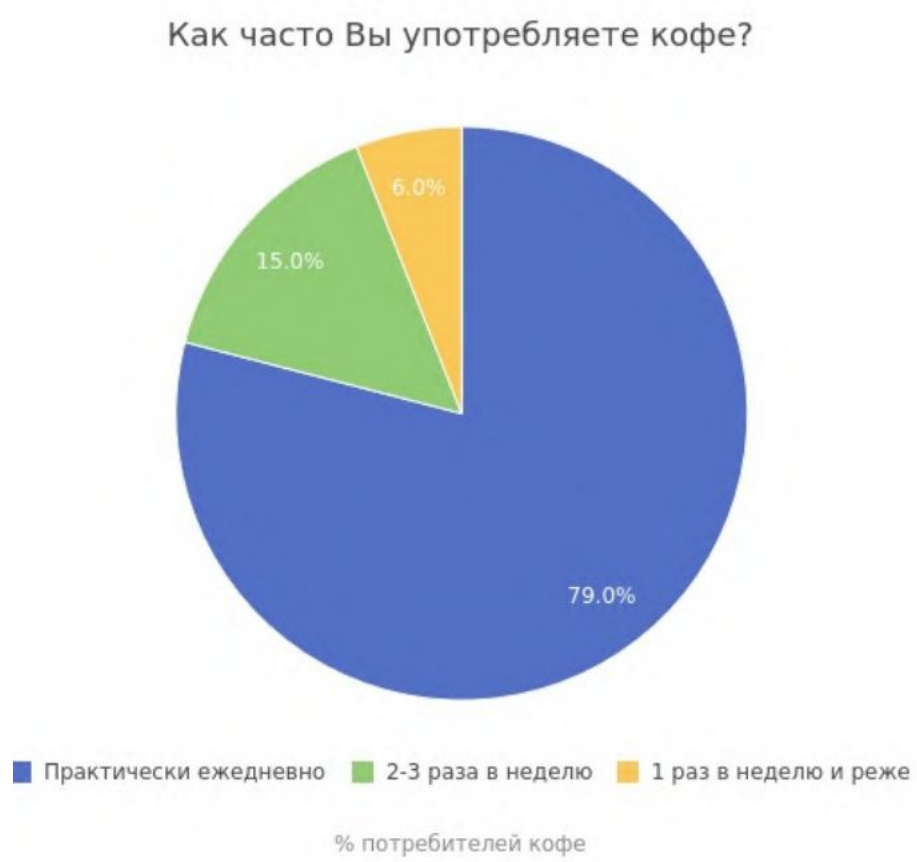
Приложение позволяет найти ближайшую кофейню с выбранным напитком. Также данное приложение выполняет поисковую функцию: позволяет пользователю просмотреть ассортимент кофеен, выбрать интересующие его напитки и сделать предзаказ выбранного товара.



Актуальность работы

Культура кофе в России стремительно развивается, подавляющая часть населения употребляет его практически ежедневно. Также, даже несмотря на значительный рост цен на кофе, люди все равно не готовы отказаться от любимого напитка.

В нынешних реалиях с большим количеством кофеен и ассортимента в них бывает трудно найти желаемый напиток поблизости, поэтому приложение «КофеСкаут» будет пользоваться спросом.



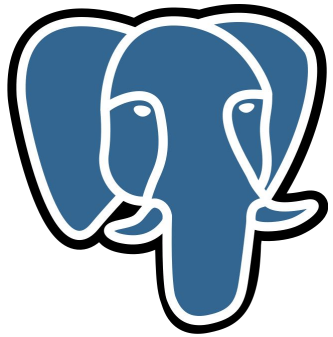
Цели

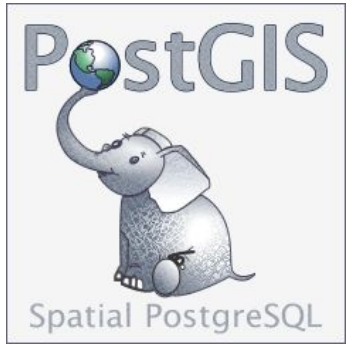
1) Разработать мобильное приложение для IOS и Android, позволяющее искать и заказывать желаемые напитки в кофейнях поблизости.
2) Разработать веб-приложение для администраторов кофеен, позволяющее редактировать меню и информацию о кофейне и отслеживать заказы.

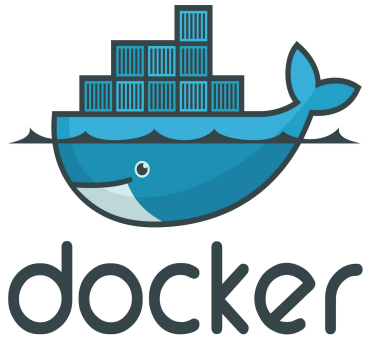
Задачи

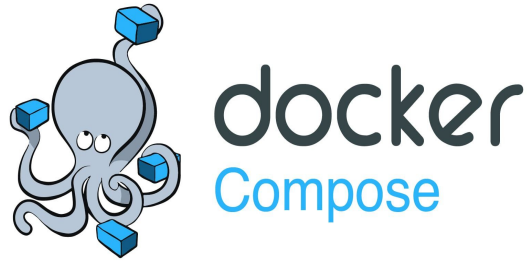
1) Выбрать стек технологий, подходящий для реализации всех функциональных требований.
2) Спроектировать архитектуру приложения CoffeeScout.
3) Выбрать архитектуру для Backend.
4) Разработать архитектуру базы данных.
5) Разработать Backend приложения в виде REST API.
6) Настроить развертывание и взаимодействие компонентов приложения в соответствии с архитектурой.

Технологический стек (backend)

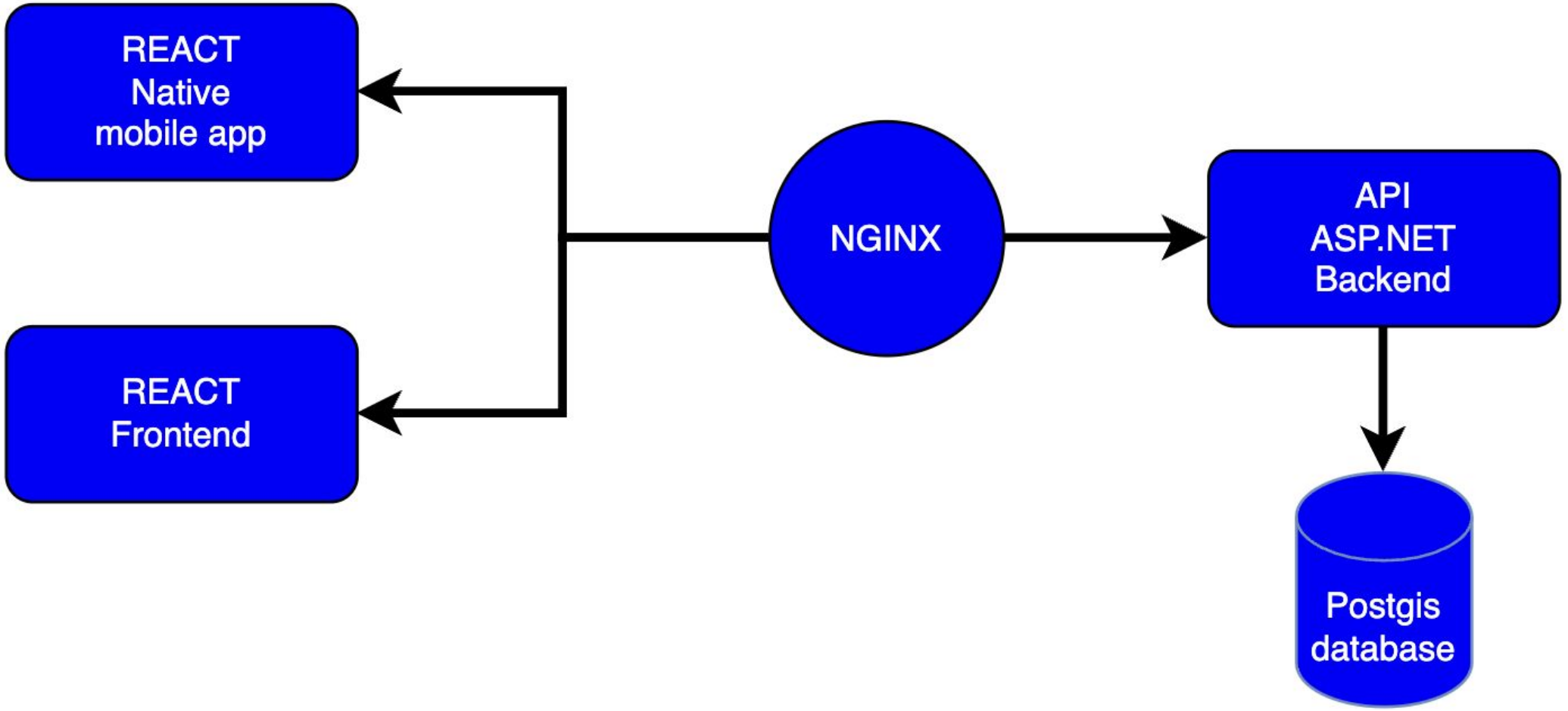




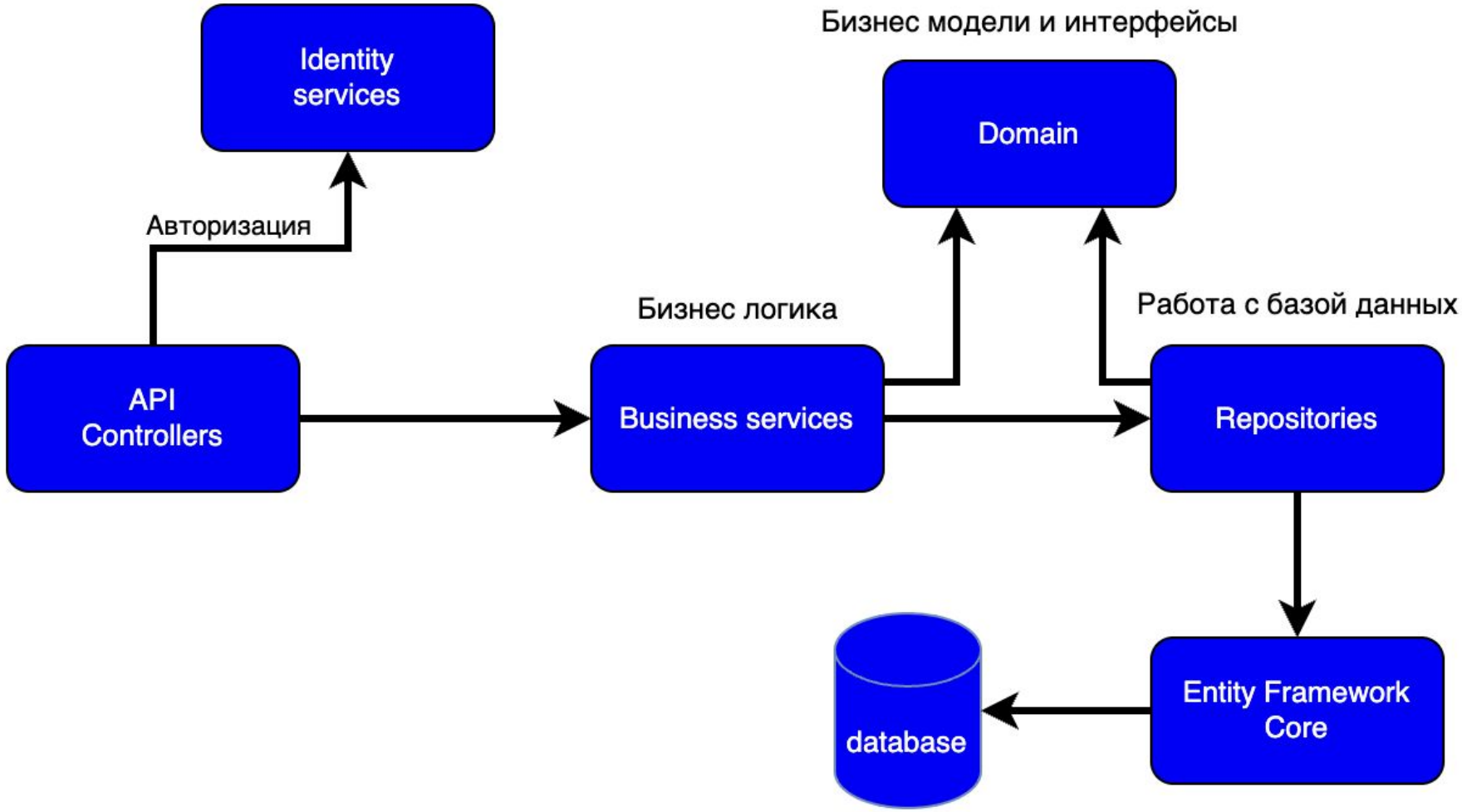




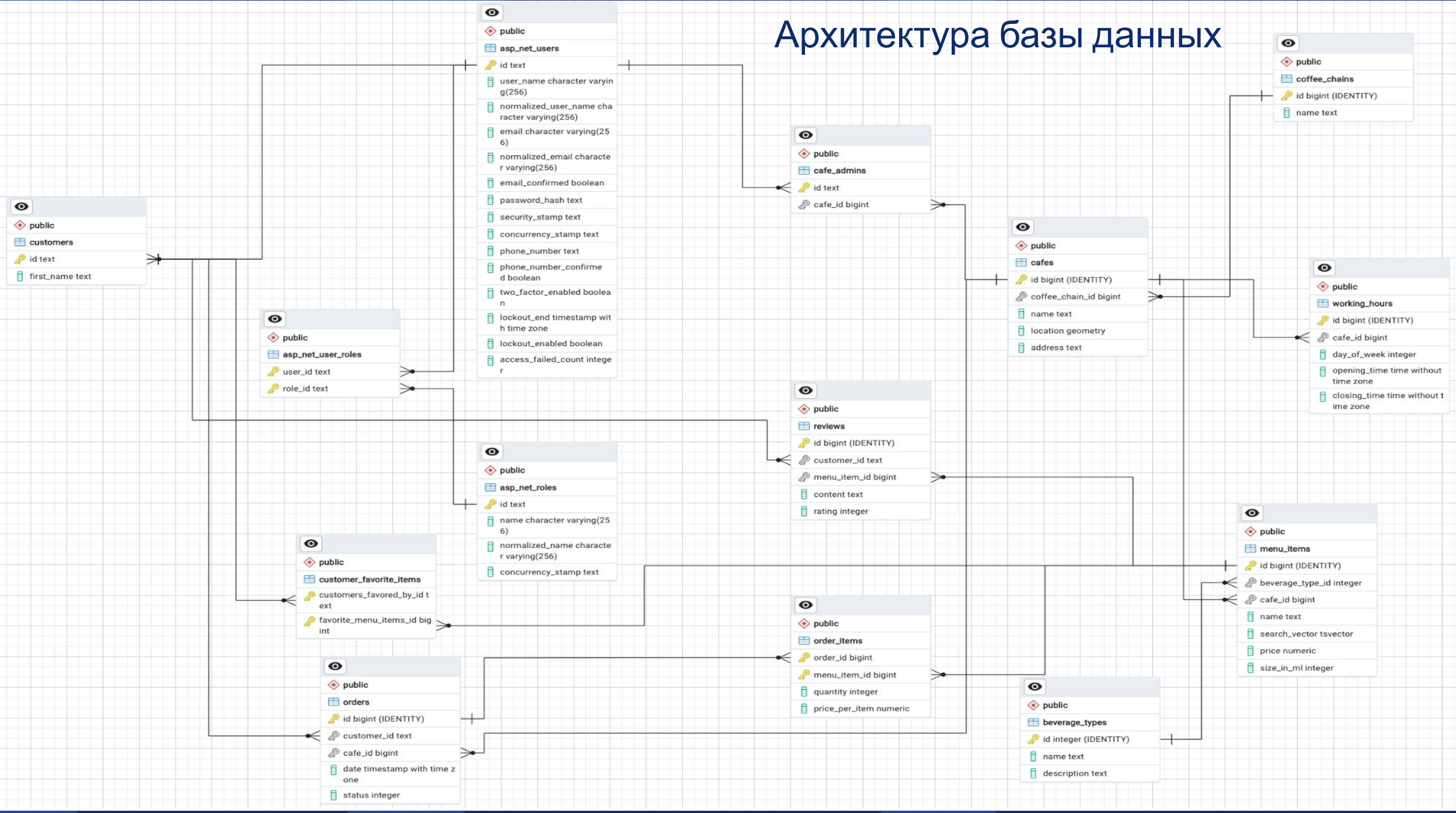
Архитектура приложения



Архитектура Backend



Архитектура базы данных



Accounts

POST	/api/v1/accounts/login
POST	/api/v1/accounts/refresh
GET	/api/v1/accounts/confirmEmail
POST	/api/v1/accounts/resendConfirmationEmail
POST	/api/v1/accounts/forgotPassword
POST	/api/v1/accounts/resetPassword
GET	/api/v1/accounts/manage/info
POST	/api/v1/accounts/manage/info
POST	/api/v1/accounts/customer/register
POST	/api/v1/accounts/cafe-admin/register

BeverageTypes

POST /api/v1/beverage-types

GET /api/v1/beverage-types

PATCH /api/v1/beverage-types/{id}

DELETE /api/v1/beverage-types/{id}

Cafes

GET	/api/v1/cafes
POST	/api/v1/cafes
PATCH	/api/v1/cafes
GET	/api/v1/cafes/info
GET	/api/v1/cafes/{id}
DELETE	/api/v1/cafes/{id}
POST	/api/v1/cafes/{id}/orders
GET	/api/v1/cafes/orders
PATCH	/api/v1/cafes/orders/{id}/complete
PATCH	/api/v1/cafes/orders/{id}/cancel
GET	/api/v1/cafes/{id}/menuItems
GET	/api/v1/cafes/menuItems

CoffeeChains

GET	/api/api/v1/coffee-chains
POST	/api/api/v1/coffee-chains
GET	/api/api/v1/coffee-chains/{id}
PATCH	/api/api/v1/coffee-chains/{id}
DELETE	/api/api/v1/coffee-chains/{id}

Customers

GET	/api/v1/customers/info
PATCH	/api/v1/customers/info
POST	/api/v1/customers/favored-menu-items
GET	/api/v1/customers/favored-menu-items
DELETE	/api/v1/customers/favored-menu-items/{menuItemId}
GET	/api/v1/customers/favored-beverage-types
GET	/api/v1/customers/orders
PATCH	/api/v1/customers/orders/{id}/pay
PATCH	/api/v1/customers/orders/{id}/cancel
PATCH	/api/v1/customers/reviews/{reviewId}
DELETE	/api/v1/customers/reviews/{reviewId}

MenuItems

GET /api/v1/menu-items

POST /api/v1/menu-items

GET /api/v1/menu-items/search

PATCH /api/v1/menu-items/{id}

DELETE /api/v1/menu-items/{id}

POST /api/v1/menu-items/{menuItemId}/reviews

GET /api/v1/menu-items/{menuItemId}/reviews

Orders

GET /api/v1/orders/{id}

PATCH /api/v1/cafes

Available to Roles CafeAdmin

Parameters Try it out

No parameters

Request body application/json

Example Value | Schema

```
{
  "name": "string",
  "location": {
    "latitude": 0,
    "longitude": 0
  },
  "address": "string",
  "workingHours": [
    {
      "dayOfWeek": 0,
      "openingTime": {
        "hour": 0,
        "minute": 0
      },
      "closingTime": {
        "hour": 0,
        "minute": 0
      }
    }
  ]
}
```

Responses

Code	Description	Links

```yaml
services:
  frontend:
    platform: linux/amd64
    image: mayaffia/coffee-scout-web
    depends_on:
      backend:
        condition: service_healthy
    restart:
      always
    networks:
      - app-network
    ports:
      - "3000:3000"
```

```yaml
backend:
  depends_on:
    database:
      condition: service_healthy
  restart: always
  networks:
    - app-network
    - backend-network
  image: doreml/coffee-scout-backend.api
  env_file:
    - .env
  ports:
    - "8000:8080"
  environment:
    - ASPNETCORE_ENVIRONMENT=Production
  healthcheck:
    test: [ "CMD", "curl", "-f", "http://localhost:8080/api/v1/health" ]
    interval: 30s
    timeout: 10s
    retries: 3
    start_period: 15s
```

```
    deploy:
      resources:
        limits:
          cpus: '2.0'
          memory: 4G
        reservations:
          cpus: '1.0'
          memory: 2G
    logging:
      driver: json-file
      options:
        max-size: "200m"
        max-file: "10"
```

```
database:
  networks:
    - backend-network
  image: postgis/postgis:16-3.4
  environment:
    POSTGRES_USER: ${DATABASESETTINGS__USERID}
    POSTGRES_PASSWORD: ${DATABASESETTINGS__PASSWORD}
    POSTGRES_DB: ${DATABASESETTINGS__DATABASE}
  ports:
    - "5432:5432"
  volumes:
    - db_data:/var/lib/postgresql/data
  healthcheck:
    test: [ "CMD", "pg_isready", "-U", "${DATABASESETTINGS__USERID}" ]
    start_period: 10s
    interval: 10s
    timeout: 5s
    retries: 5
  deploy:
    resources:
      limits:
        cpus: '2.0'
        memory: 1500M
      reservations:
        cpus: '1.0'
        memory: 1000M
```

```yaml
  nginx:
    networks:
      - app-network
    image: nginx
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      backend:
        condition: service_healthy

networks:
  app-network:
    driver: bridge
  backend-network:
    driver: bridge

volumes:
  db_data:
    driver: local
```

Deployment

Дальнейшее развитие

1)Реализовать систему push-уведомлений.

2)Реализовать систему сбора статистики и метрик.

3)Реализовать систему рекомендаций напитков.

4)Увеличить процент покрытия тестами кодовой базы.

5)Реализовать взаимодействие систем по https с использованием TSL/SSL сертификатов.

6)Улучшение пользовательского интерфейса.

7)Оптимизировать скорость загрузки страниц.

8)Разработать Web версию для суперадминистратора.

9)Реализовать систему генерирования уникальных кодов для регистрации кофеен.

10)Интегрировать реальную платежную систему.

.

Список использованных источников

- https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures
- https://www.lucidchart.com/pages/er-diagrams
- https://swagger.io/
- https://docs.docker.com/compose/production/

Спасибо за внимание!