Дисциплина «Backend-разработка: Web API»

Куратор: SimbirSoft

Здравствуй, друг! Мы рады приветствовать тебя на полуфинальном этапе олимпиады «Волга-IT. Ты можешь выбрать для работы язык программирования C#, Java, Python или GO.

Тебе предстоит окунуться в разработку инновационной системы для клиник под названием Simbir.Health. Это платформа для управления клиническими процессами, нацеленная на революцию в доступе к медицинским данным и эффективности работы медицинских учреждений. Система позволяет регистрировать новых пациентов, врачей, создавать и обновлять расписание приёмов, а также управлять медицинскими записями и документацией. Особое внимание в проекте уделяется безопасности и конфиденциальности данных, а также возможности масштабирования системы для работы с большим количеством клиник.

Мы уверены, что ты сможешь проявить себя и внести важный вклад в разработку системы, которая в будущем поможет многим клиникам оптимизировать их работу и повысить качество обслуживания пациентов. Желаем удачи в реализации задания и надеемся, что твоё участие станет важным шагом на пути к твоему профессиональному развитию!

ОБЩЕЕ ОПИСАНИЕ ЗАДАЧИ

В данном задании вы будете работать над созданием микросервисного приложения, которое охватывает различные аспекты программной инженерии, включая разработку микросервисов, конфигурацию API, использование баз данных и многое другое. Ваша цель — разработать ряд микросервисов, обеспечивающих функциональность для моделирования работы больницы, а также реализовать дополнительные задачи для расширенной функциональности и интеграции.

АРХИТЕКТУРА ПРИЛОЖЕНИЯ

Account microservice отвечает за авторизацию и данные о пользователях. Все остальные сервисы зависят от него, ведь именно он выпускает JWT токен и проводит интроспекцию.

Hospital microservice отвечает за данные о больницах, подключенных к системе. Отправляет запросы в микросервис аккаунтов для интроспекции токена.

Timetable microservice отвечает за расписание врачей и больниц, а также за запись на приём пользователем. Отправляет запросы в микросервис аккаунтов для интроспекции токена и проверки существования связанных сущностей. Отправляет запросы в микросервис больниц для проверки существования связанных сущностей.

Document microservice отвечает за историю посещений пользователя. Отправляет запросы в микросервис аккаунтов

для интроспекции токена и проверки существования связанных сущностей. Отправляет запросы в микросервис больниц для проверки существования связанных сущностей.

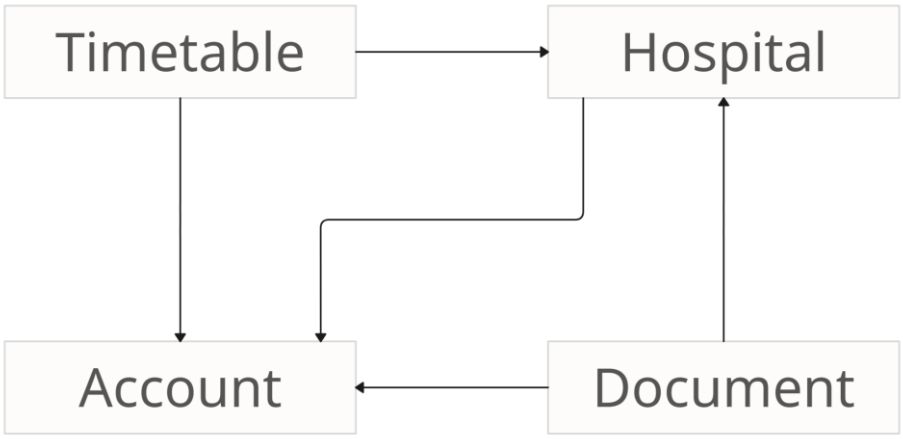

Рис.1. Диаграмма зависимостей микросервисов друг от друга.

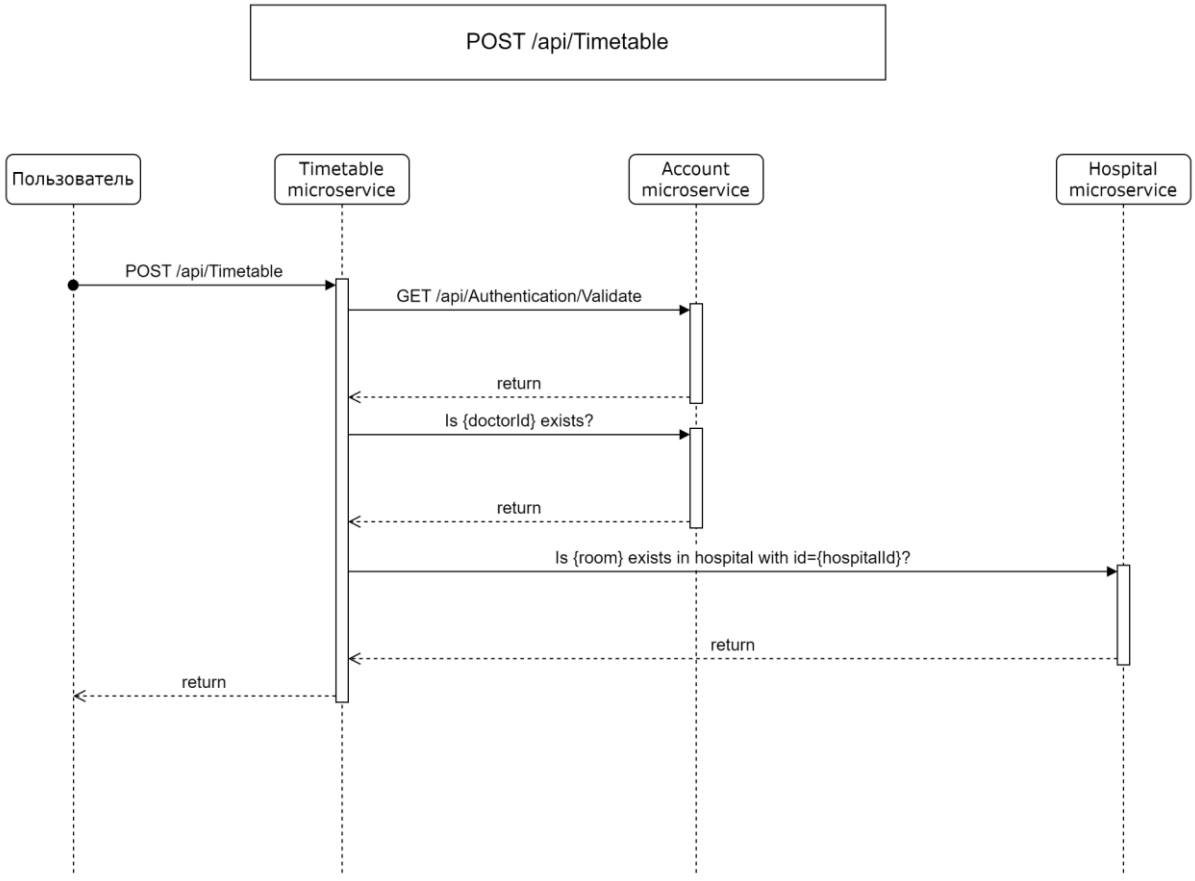

Рис.2. Пример межсервисных запросов в api.

РАЗРАБОТКА ПРИЛОЖЕНИЯ

Порядок выполнения разработки основного задания:

1. Необходимо разработать микросервис аккаунтов.
2. Необходимо разработать микросервис больниц.
3. Необходимо разработать микросервис расписания.
4. Необходимо разработать микросервис документов.

Ограничения основного задания:

1. Используйте базу данных PostgreSQL.
2. Для авторизации между сервисами используйте JWT.
3. Выберите один из методов взаимодействия микросервисов на свое усмотрение. Доступные методы: HTTP, gRPC, RabbitMQ или Apache Kafka.
4. Все API должны быть задокументированы с помощью Swagger с возможностью авторизации через JWT, ссылки на Swagger разместите в файле README.md.
5. При реализации сервера авторизации вы не должны использовать готовые решения по типу (identity server, keycloak и т.п.).
6. При выполнении команды “docker-compose up -d” ваши микросервисы не должны скачиваться с docker-hub, а должны билдиться из исходников.

Порядок выполнения дополнительных заданий:

1. Использовать Elasticsearch при поиске в документах.

Ограничения дополнительных заданий:

1. Ссылки Elasticsearch и Kibana разместите в файле README.md.

РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЯ

Приложение должно быть контейнеризировано и запускаться через Docker. При проверке мы будем запускать единственную команду и ожидаем что все сервисы будут иметь предустановленные настройки и пользователей. Команда запуска: docker-compose up -d

Предустановленные аккаунты по умолчанию:

№	Логин	Пароль	Роль
1	admin	admin	Admin
2	manager	manager	Manager
3	doctor	doctor	Doctor
4	user	user	User

ОТПРАВКА ЗАДАНИЯ

1. Выполните основную часть задания.
2. Оформите документацию в файле README.md с инструкциями по запуску.
3. Разместите ваш проект на одной из платформ: Github, Gitlab, Bitbucket, GitVerse.
4. Отправьте ссылку на проект через личный кабинет для проверки.

ОТВЕТЫ НА ВОПРОСЫ

Если у вас возникли вопросы по заданию, то решите его на свое усмотрение, а мы оценим ваше решение. Если ваше решение отличается от задания, то обязательно напишите все изменения в файл Readme.md.

Пример README.md

```
# Основное задание:
1. Account URL: http://localhost:8081/ui-swagger
2. Hospital URL: http://localhost:8082/ui-swagger
3. Timetable URL: http://localhost:8083/ui-swagger
4. Document URL: http://localhost:8084/ui-swagger

# Дополнительное задание:
1. ElasticSearch URL: http://elasticsearch-service/
2. Kibana URL: http://kibana-service/

# Дополнительная информация которую вы захотите указать
...
```

МИКРОСЕРВИС АККАУНТОВ

POST /api/Authentication/SignUp

описание: Регистрация нового аккаунта
body:
```
{
  “lastName”: "string",
  “firstName”: "string",
  "username": "string",
  "password": "string"
}
```
ограничения: Нет

POST /api/Authentication/SignIn

описание: Получение новой пары jwt пользователя
body:
```
{
  "username": "string",
  "password": "string"
}
```
ограничения: Нет

PUT /api/Authentication/SignOut

описание: Выход из аккаунта
ограничения: Только авторизованные пользователи

GET /api/Authentication/Validate

описание: Интроспекция токена
параметры:
accessToken: string
ограничения: Нет

POST /api/Authentication/Refresh

описание: Обновление пары токенов
body:
```
{
  “refreshToken”: “string”
}
```
ограничения: Нет

GET /api/Accounts/Me

описание: Получение данных о текущем аккаунте
ограничения: Только авторизованные пользователи

PUT /api/Accounts/Update

описание: Обновление своего аккаунта
body:

```
{
  “lastName”: "string",
  “firstName”: "string",
  "password": "string"
}
```

ограничения: Только авторизованные пользователи

GET /api/Accounts

описание: Получение списка всех аккаунтов
параметры:

```
from: int       //Начало выборки
count: int      //Размер выборки
```

ограничения: Только администраторы

POST /api/Accounts

описание: Создание администратором нового аккаунта
body:

```
{
  “lastName”: "string",
  “firstName”: "string",
  "username": "string",                 //имя пользователя
  "password": "string",                 //пароль
  "roles": [
    "string"                        //массив ролей пользователя
  ]
}
```

ограничения: Только администраторы

PUT /api/Accounts/{id}

описание: Изменение администратором аккаунта по id
body:

```
{
  “lastName”: "string",
  “firstName”: "string",
  "username": "string",                 //имя пользователя
  "password": "string",                 //пароль
  "roles": [
    "string"                            //массив ролей пользователя
  ]
}
```

ограничения: Только администраторы

DELETE /api/Accounts/{id}

описание: Мягкое удаление аккаунта по id
ограничения: Только администраторы

GET /api/Doctors

описание: Получение списка докторов
параметры:
nameFilter: string //Фильтр имени *(FullName LIKE ‘%{nameFilter}%’)*
from: int //Начало выборки
count: int //Размер выборки
ограничения: Только авторизованные пользователи

GET /api/Doctors/{id}

описание: Получение информации о докторе по Id
ограничения: Только авторизованные пользователи

МИКРОСЕРВИС БОЛЬНИЦ

GET /api/Hospitals

описание: Получение списка больниц
параметры:
from: int //Начало выборки
count: int //Размер выборки
ограничения: Только авторизованные пользователи

GET /api/Hospitals/{id}

описание: Получение информации о больнице по Id
ограничения: Только авторизованные пользователи

GET /api/Hospitals/{id}/Rooms

описание: Получение списка кабинетов больницы по Id
ограничения: Только авторизованные пользователи

POST /api/Hospitals

описание: Создание записи о новой больнице
body:

```
{
  “name”: “string”,
  “address”: “string”,
  “contactPhone”: “string”,
  “rooms”: [
    “string”                    //массив наименований кабинетов
  ]
}
```

ограничения: Только администраторы

PUT /api/Hospitals/{id}

описание: Изменение информации о больнице по Id

body:

```
{
  “name”: “string”,
  “address”: “string”,
  “contactPhone”: “string”,
  “rooms”: [
    “string”                    //массив наименований кабинетов
  ]
}
```

ограничения: Только администраторы

DELETE /api/Hospitals/{id}

описание: Мягкое удаление записи о больнице
ограничения: Только администраторы

МИКРОСЕРВИС РАСПИСАНИЯ

POST /api/Timetable

описание: Создание новой записи в расписании
body:
{
“hospitalId”: int,
"doctorId": int,
"from": "dateTime(ISO8601)",
"to": "dateTime(ISO8601)",
"room": "string"
}
ограничения: Только администраторы и менеджеры. {from} и {to} - количество минут всегда кратно 30, секунды всегда 0 (пример: “2024-04-25T11:30:00Z”, “2024-04-25T12:00:00Z”). {to} > {from}. Разница между {to} и {from} не должна превышать 12 часов.

PUT /api/Timetable/{id}

описание: Обновление записи расписания
body:
{
"hospitalId": int,
"doctorId": int,
"from": "dateTime(ISO8601)",
"to": "dateTime(ISO8601)",
"room": "string"
}
ограничения: Только администраторы и менеджеры. Нельзя изменить если есть записавшиеся на прием. {from} и {to} - количество минут всегда кратно 30, секунды всегда 0 (пример: “2024-04-25T11:30:00Z”, “2024-04-25T12:00:00Z”). {to} > {from}. Разница между {to} и {from} не должна превышать 12 часов.

DELETE /api/Timetable/{id}

описание: Удаление записи расписания
ограничения: Только администраторы и менеджеры

DELETE /api/Timetable/Doctor/{id}

описание: Удаление записей расписания доктора
ограничения: Только администраторы и менеджеры

DELETE /api/Timetable/Hospital/{id}

описание: Удаление записей расписания больницы
ограничения: Только администраторы и менеджеры

GET /api/Timetable/Hospital/{id}

описание: Получение расписания больницы по Id
параметры:
from: string(ISO8601)
to: string(ISO8601)
ограничения: Только авторизованные пользователи

GET /api/Timetable/Doctor/{id}

описание: Получение расписания врача по Id
параметры:
from: string(ISO8601)
to: string(ISO8601)
ограничения: Только авторизованные пользователи

GET /api/Timetable/Hospital/{id}/Room/{room}

описание: Получение расписания кабинета больницы
параметры:
from: string(ISO8601)
to: string(ISO8601)
ограничения: Только администраторы и менеджеры и врачи

GET /api/Timetable/{id}/Appointments

описание: Получение свободных талонов на приём.
детали: Каждые 30 минут из записи расписания - это один талон. Если в сущности Timetable from=2024-04-25T11:00:00Z, to=2024-04-25T12:30:00Z, то

запись доступна на: 2024-04-25T11:00:00Z, 2024-04-25T11:30:00Z, 2024-04-25T12:00:00Z.
ограничения: Только авторизованные пользователи

POST /api/Timetable/{id}/Appointments

описание: Записаться на приём
body:
{
  “time”: “dateTime(ISO8601)”
}
ограничения: Только авторизованные пользователи

DELETE /api/Appointment/{id}

описание: Отменить запись на приём
ограничения: Только администраторы, менеджеры, и записавшийся пользователь

МИКРОСЕРВИС ДОКУМЕНТОВ

GET /api/History/Account/{id}

описание: Получение истории посещений и назначений аккаунта
детали: Возвращает записи где {pacientId}={id}
ограничения: Только врачи и аккаунт, которому принадлежит история

GET /api/History/{id}

описание: Получение подробной информации о посещении и назначениях
ограничения: Только врачи и аккаунт, которому принадлежит история

POST /api/History

описание: Создание истории посещения и назначения
body:

```
{
  “date”: “dateTime(ISO8601)”,
  “pacientId”: int,
  “hospitalId”: int,
  “doctorId”: int,
  "room": "string",
  “data”: “string”
}
```

ограничения: Только администраторы и менеджеры и врачи. {pacientId} - с ролью User.

PUT /api/History/{id}

описание: Обновление истории посещения и назначения
body:

```
{
  “date”: “dateTime(ISO8601)”,
  “pacientId”: int,
  “hospitalId”: int,
  “doctorId”: int,
  "room": "string",
  “data”: “string”
}
```

ограничения: Только администраторы и менеджеры и врачи. {pacientId} - с ролью User.