Санкт-Петербургский Политехнический Университет Петра Великого
Институт Компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет

Предмет: Программное обеспечение распределённых вычислительных систем

Тема: Разработка системы "Дополнительное образование"

Студент:___Иванов А.А._____
Гр. № ____3540901/91502____

Преподаватель: Стручков И.В.

Санкт-Петербург
2020

Оглавление

1.Постановка задачи

1.1. Назначение проектируемой системы

Разрабатываемая система служит для организации услуг в сфере дополнительного образования и в рамках отдельно взятой компании по предоставлению данных услуг.

1.2. Функциональные требования

Администратор – сотрудник компании, занимается взаимодействием с клиентами, составлением расписания, жалобами от учеников, урегулированием конфликтов и неточностей, а так же финансами.

Учитель – сотрудник компании, занимается непосредственным оказанием образовательных услуг клиенту.

Клиент – ученик, или представитель ученики, может записаться на урок, подать жалобу, оспорить прогул в случае заморозки абонимента или болезни, попросить назначить дополнительное занятие.

2.Бизнесс процессы

2.1. Формирование расписания

1. Учитель выбирает дни, в которые может провести занятия
2. Клиент оставляет заявку на занятие и время, в которое он может их посещать
3. Администратор формирует расписание и создает уведомления для учителя и клиентов
4. Учитель подтверждает готовность проводить занятие в конкретный день недели
5. Клиент подтверждает готовность посещать занятие в конкретный день недели
6. После проведения занятия учитель закрывает занятие, как проведенное
7. Клиенту может посмотреть результаты проведенного урока, комментарий об уроке, свой статус на нем и комментарий о себе
8. Клиент может оспорить свой статус или подать жалобу на учителя
9. Администратор разрешает оба типа конфликта в случае возникновения

Первый тип конфликта - оспорение статуса. В случае прогула с клиента снимают деньги как за проведенное занятие. В случае пропуска по предупреждению заранее или уважительной причине (на взгляд администратора) администратор переносит занятие и деньги не списываются

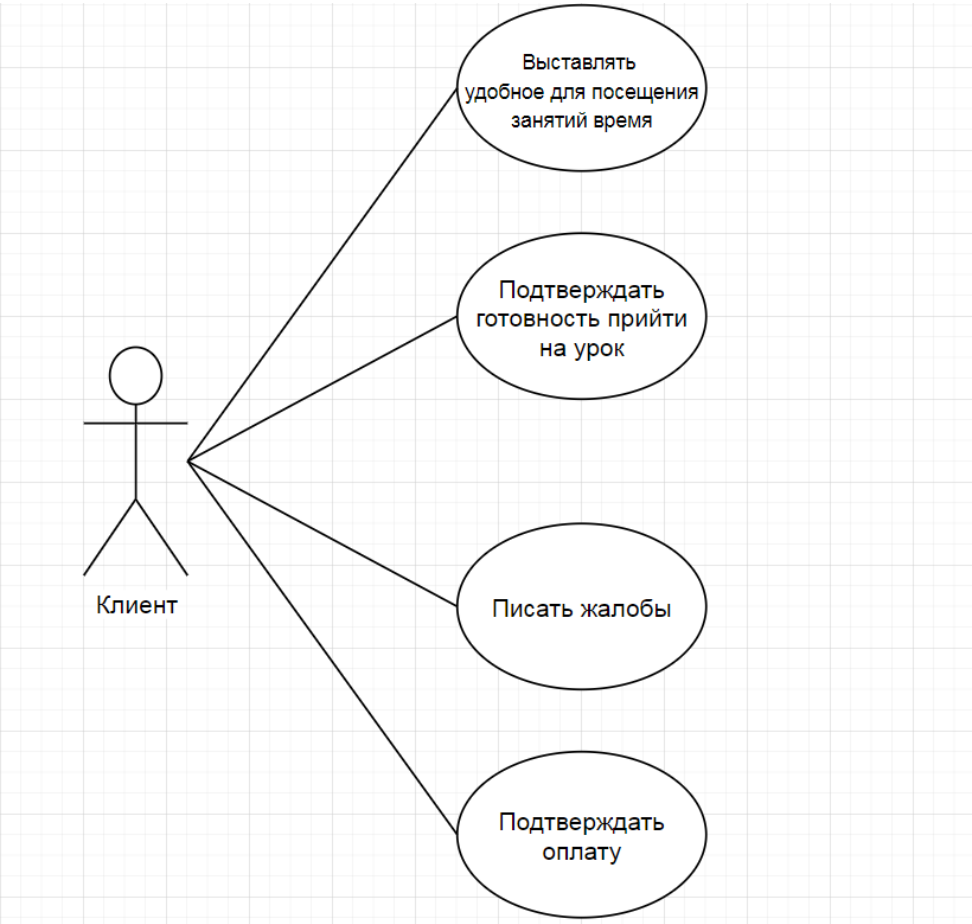
Второй тип - все остальные виды жалоб от клиента. Разрешаются администратором в индивидуальном порядке

2.2. Оплата занятий/заработная плата

- За каждое закрытое преподавателем занятие ему начисляется плата
- За каждое поещенное/прогулянное занятие с клиента снимаются деньги
- В конце отчетного периода администратор формирует уведомление об оплате/зарплате
- Клиент может подтвердить чек или опровергнуть
- В случае подтверждения с клиента списываются деньги, преподавателю начисляется
- В случае опровержения администратор связывается с клиентом и может внести изменения в закрытые занятия

3.Разработка вариантов использования

3.1. Клиент



Запись в учебную группу

- **Клиент** отправляет запрос на запись в группу, где указывает все варианты удобного времени для посещения занятий
- **Клиент** получает уведомление о записи его в группу с указанием времени и дня недели, выделенного для занятий.

- **Клиент** подтверждает запись

Альтернатива **Клиент** передумал записываться или время перестало быть удобным. В таком случае он отклоняет уведомление и начинает процесс записи с начала.

Реакция на завершившееся занятие

1. **Клиент** не доволен преподавателем или на уроке случился инцидент.
2. **Клиент** заходит на страницу с занятием и пишет жалобу.

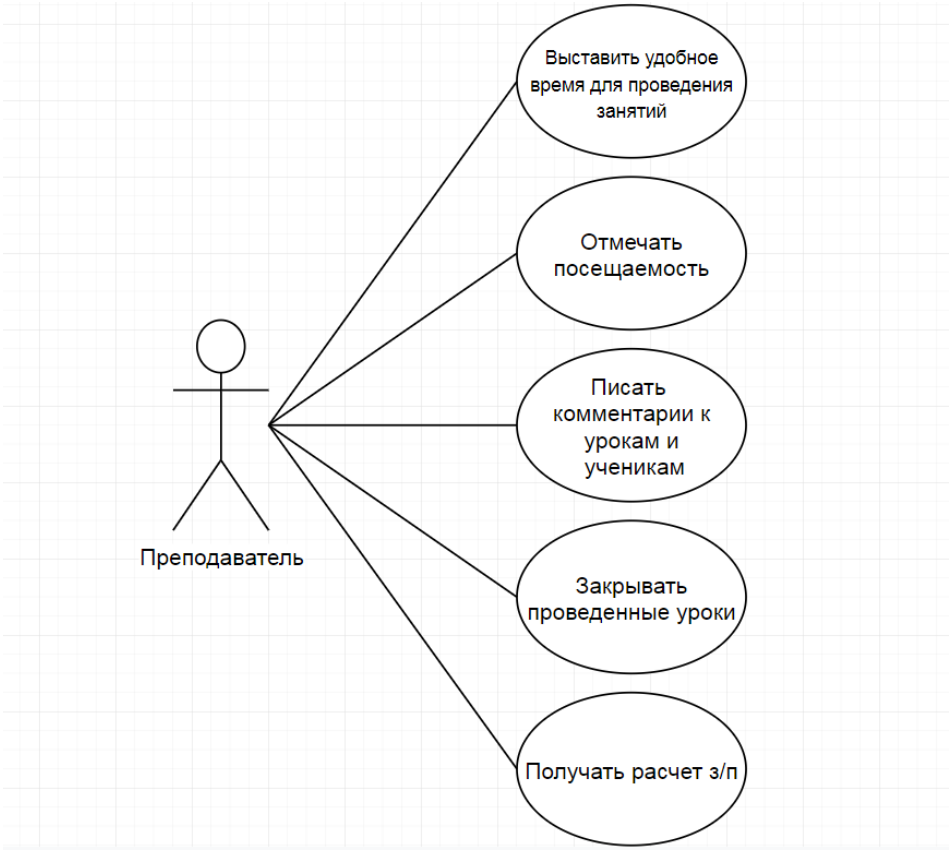
Альтернатива Претензии отсутствуют. В таком случае все пункты опускаются

Подтверждение оплаты

1. **Клиент** получает уведомление о списании средств за отчетный период
2. **Клиент** подтверждает сумму оплаты.

Альтернатива Если клиент не согласен с суммой он отклоняет подтверждение и ждет, когда с ним свяжется администратор для урегулирования.

3.2. Преподаватель



Согласование времени занятий

1. **Преподаватель** составляет свой рабочий график, где указывает все варианты удобного времени для проведения занятий
2. **Преподаватель** получает уведомление о создании учебных групп, выделенных ему, с указанием времени и дня недели, выделенного для занятий.
3. **Преподаватель** подтверждает время проведения им занятий (Данный процесс происходит один раз при создании группы. Далее урок считается регулярным и для данных учеников с данным преподавателем проводится каждую неделю)

Альтернатива Преподаватель отказывается брать учебную группу. В таком случае группа предлагается другому преподавателю, либо распускается и пересобирается в другое время.

Закрытие урока

1. **Преподаватель** после проведенного урока выбирает в расписании урок, который он провел.
2. **Преподаватель** отмечает присутствующих и отсутствующих
3. **Преподаватель** пишет комментарии к ученикам и к уроку в целом
4. **Преподаватель** закрывает урок. Уведомление о закрытом уроке направляется администратору.

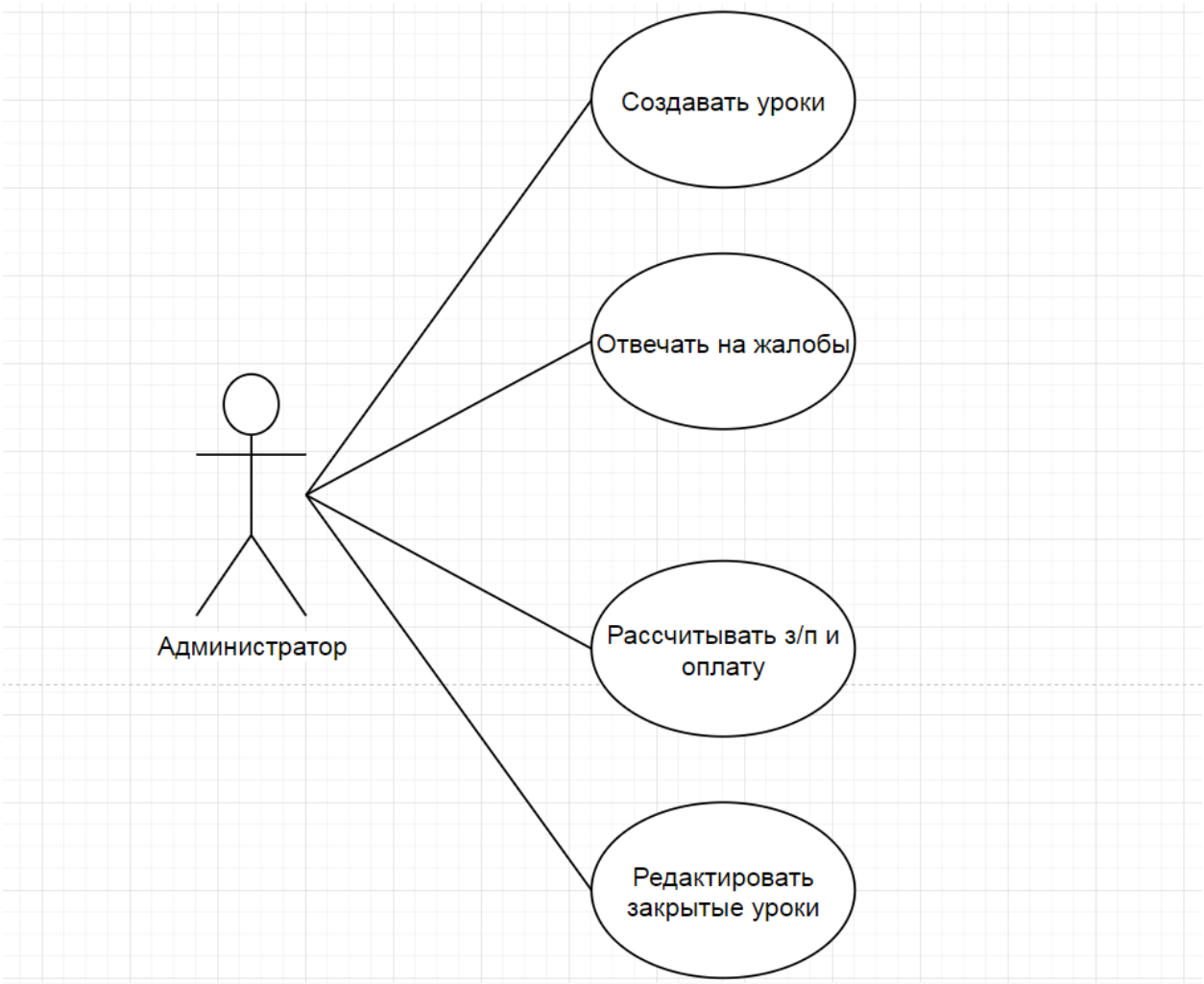
Альтернатива отсутствие комментариев по уроку или ученикам. Шаг 3 пропускается.

Подтверждение заработной платы

1. **Преподаватель** получает уведомление о начислении средств за отчетный период
2. **Преподаватель** подтверждает сумму заработной платы.

Альтернатива Если Преподаватель не согласен с суммой он отклоняет подтверждение и ждет, когда с ним свяжется администратор для урегулирования.

3.3. Администратор



Составление расписания

1. **Администратор** анализирует пришедшие заявки от клиентов и преподавателей
2. **Администратор** Соспоставляет желаемое время для преподавателя и для клиентов и формирует урок.
3. **Администратор** Посылает всем потенциальным участникам урока подтверждение
4. **Администратор** Добавляет новый урок в расписание

Альтернатива В случае отказа одного или нескольких учеников, Администратор по своему усмотрению либо убирает отказавшихся и все равно добавляет урок в расписание, либо распускает всех оставшихся участников

Альтернатива В случае отказа Преподавателя, Администратор предлагает проведение данного урока другому преподавателю, либо распускает для формирования из распущенных учеников новых уроков.

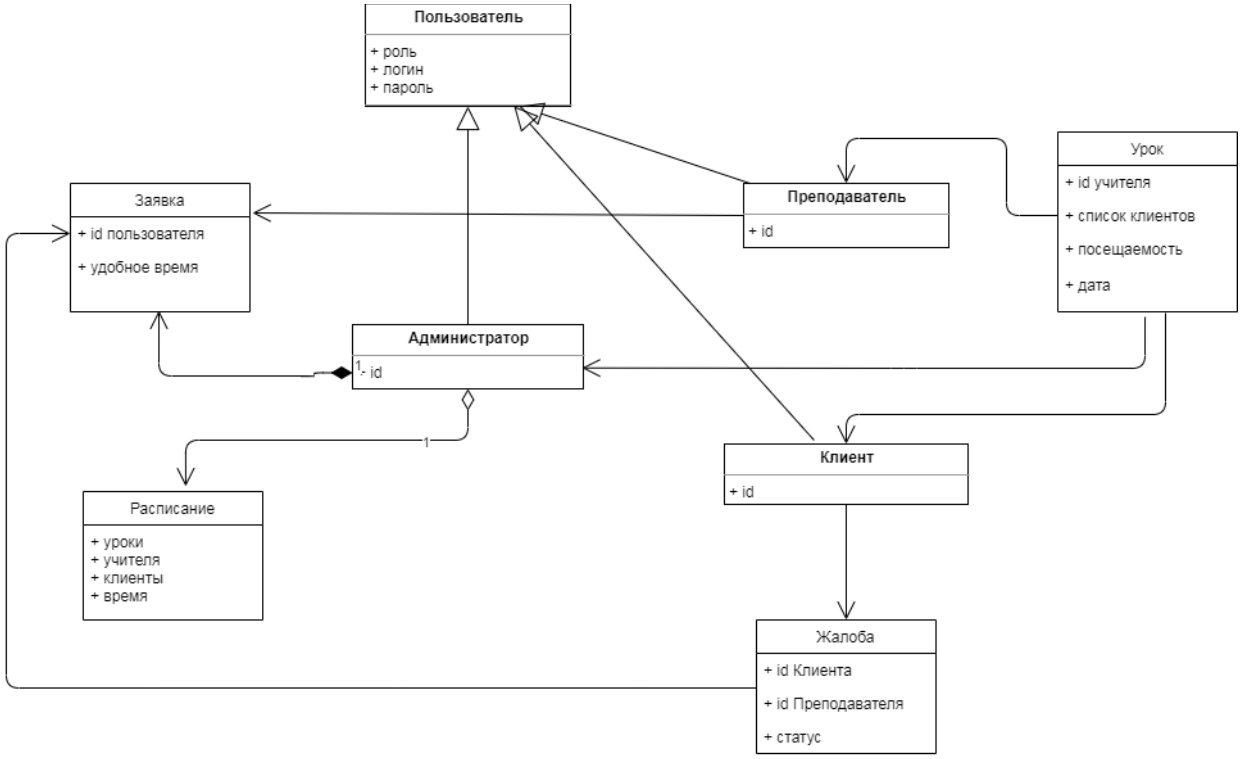
Урегулирование жалоб и несогласий со списанием\начислением

1. **Администратор** получает уведомление о жалобе\несогласии
2. **Администратор** связывается со сторонами конфликта

3. **Администратор** по своему усмотрению урегулирует вопрос, использую возможность редактирования закрытых уроков для изменения статуса ученика.

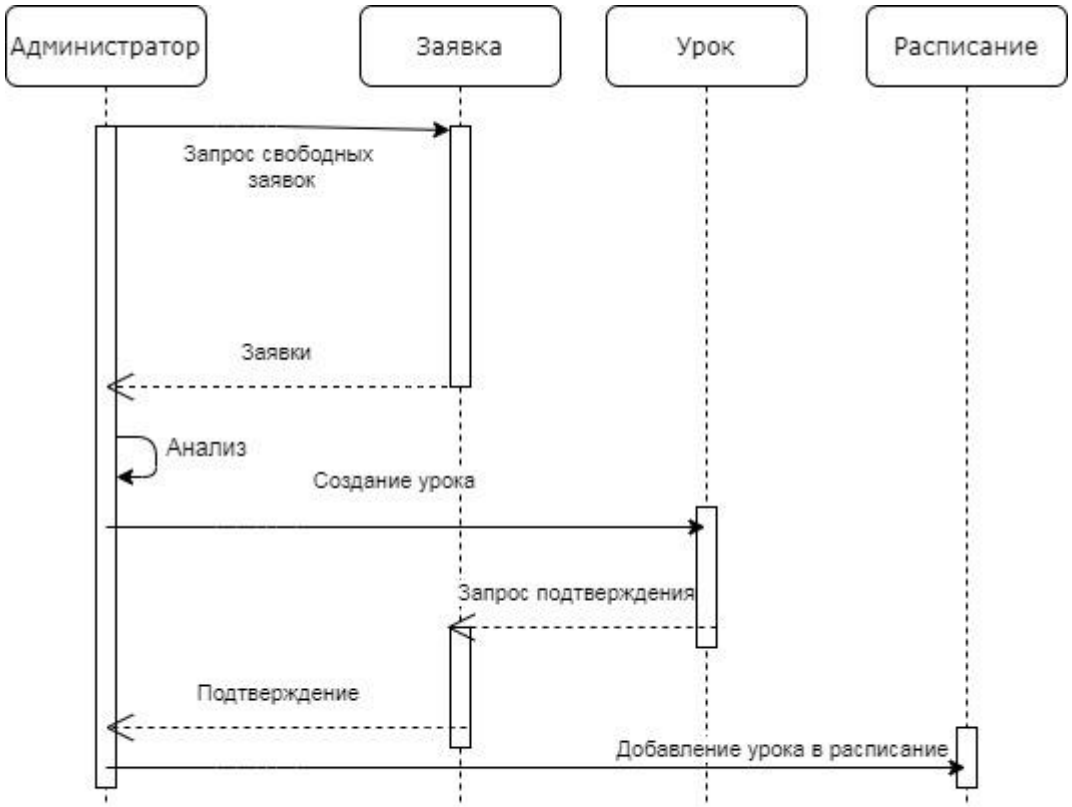
Альтернатива В случае отсутствия жалоб данные шаги пропускаются

4. Диаграмма классов

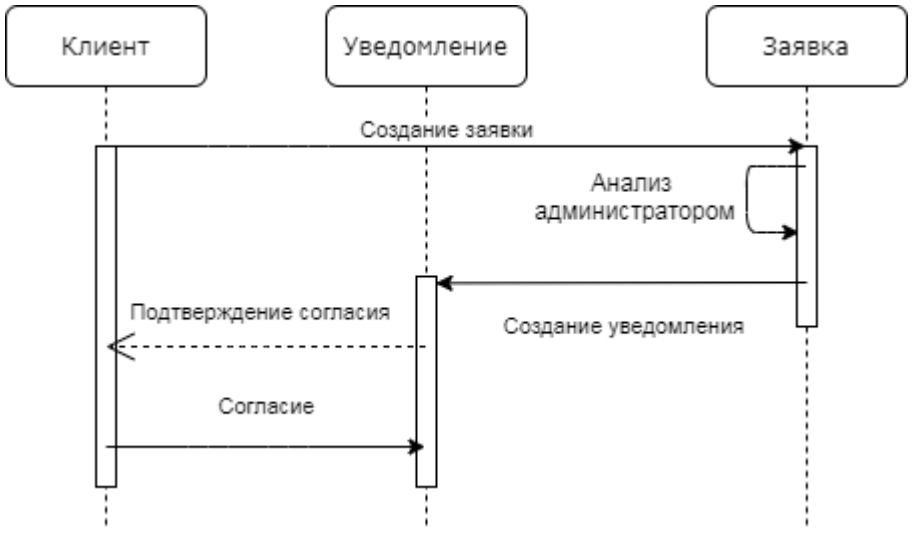


5. Диаграммы последовательностей

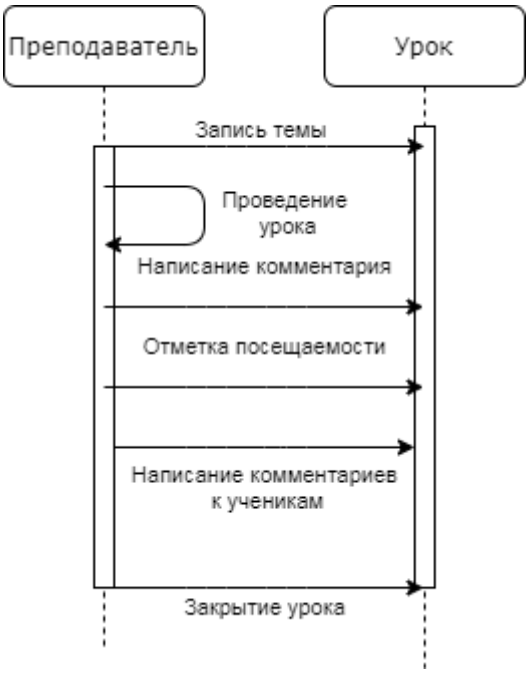
5.1. Составление расписания



5.2. Подача заявки

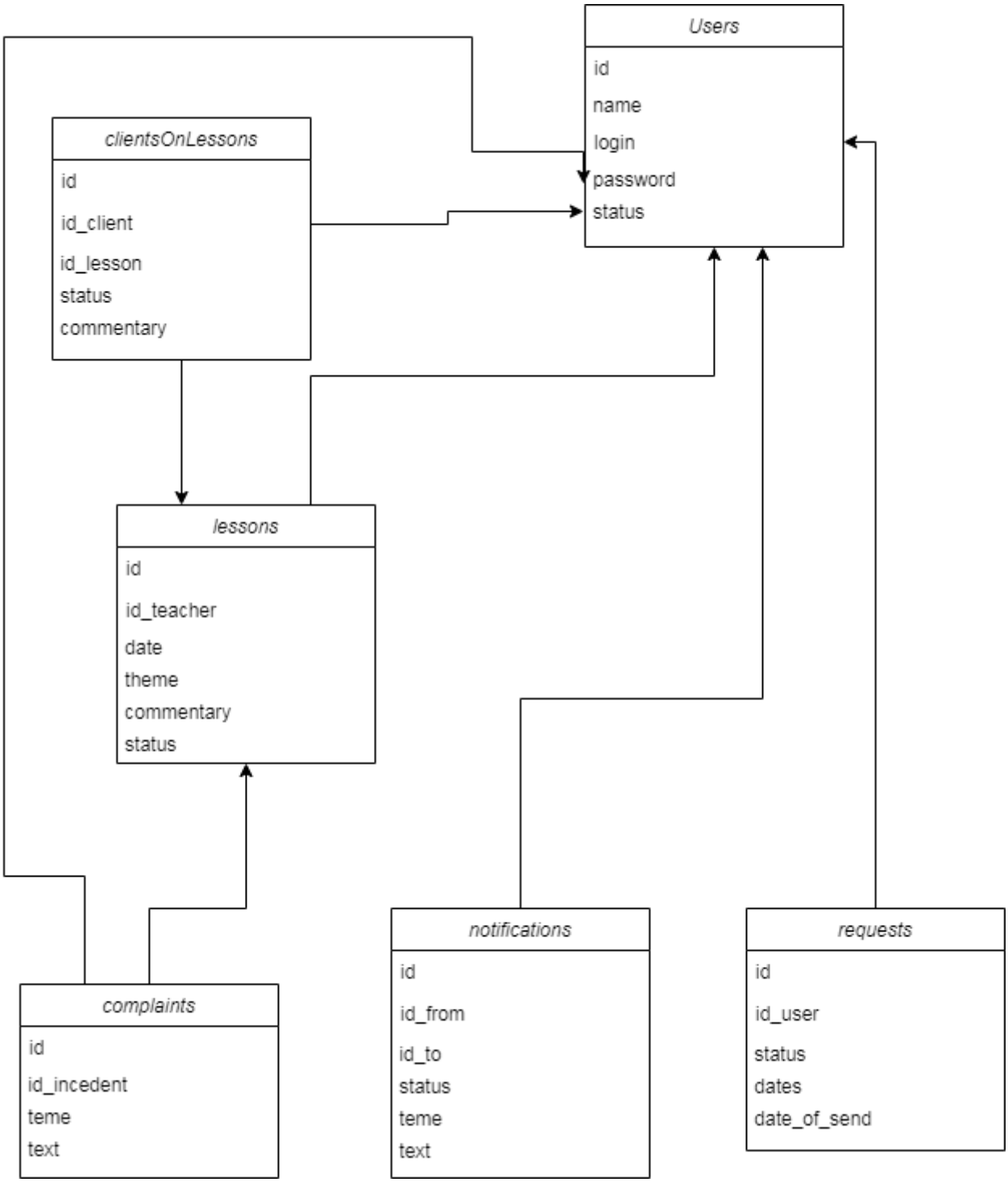


5.3. Закрытие урока преподавателем



В качестве СУБД была выбрана PostgreSQL ввиду её популярности и хорошего документирования

Схемы базы данных представлена на изображении:



6.Описание программы

6.1. Backend

Для реализации бекенда, поставленной задачи был использован Spring Framework. Прежде всего были описаны модели, которые приводились выше. Отличительной чертой в реализации следует отметить наследование от базового класса AbstractEntity, который содержит в себе только уникальный идентификатор. Все модели расширяют

данный класс, следовательно у каждой модели есть ID, по которому можно осуществлять поиск в базе данных.

Для взаимодействия с базой данных был написан слой репозиториев. Каждый репозиторий расширяет базовый интерфейс CommonRepository, который в свою очередь расширяет CrudRepository. Это сделано для того, что для каждого репозитория были доступны CRUD-методы.

Над слоем репозиториев был написан слой сервисов. В сервисах написана вся бизнес- логика данной системы. В каждом сервисе есть методы, которые описывают действия, которые могут быть выполнены конкретной сущностью.

Над слоем сервисов был написан слой REST-контролеров. В данном слое, также была вынесена асбтракция базового контролера, для возможности использования CRUD-методов. В каждом контролере присутствуют методы, которые являются endpoint приложения. В реализации данных методов вызываются соответствующие методы из слоя сервисов.

6.2. Frontend

Для реализации фронтенда, поставленной задачи был использован Angular. Было создано приложение, состоящая из компонентов. Каждый компонент представляет из себя совокупность файлов верстки, стилей и бизнес-логики, написанной на языке Typescript. Каждый компонент является отдельным экраном и инкапсулирует в себе соответствующую логику. Также в данном приложение были написаны сервисы, которые посылают запросы и принимают ответы с соответствующих endpoints бекенда. Данное приложение взаимодействует с приложением бекенда.

7.Методика и результаты тестирования

7.1. Ручное тестирование

Было проведено ручное тестирование двух частей системы: бекенда и фронтенда.

7.1.1. Backend

В качестве ручного тестирования со стороны бекенда были выполнены все возможные запросы с REST- клиента. В данных запросах были заданы соответствующие endpoint и заполнены необходимые параметры. В результате были получены ожидаемые ответы, что показывает верную работу севрверной части системы.

7.1.2. Frontend

В качестве ручного тестирования со стороны фронтенда были выполнены все возможные сценарии для каждого пользователя, используя пользовательский интерфейс. Все сценарии были успешно завершены, тем самым подтверждая корректную работу всей системы.

8.Инструкция системному администратору по развёртыванию приложения

Для развертывания данной системы необходимо наличие любой операционной системы, например Windows/Linux/MacOS. Далее перечислены все средства, требуемые к установки для развёртывания системы локально на машине:

Gradle - обычная установка, согласно прилагающейся инструкции

Postgres - обычная установка, согласно прилагающейся инструкции, версия 9.5 и выше

Java - обычная установка, согласно прилагающейся инструкции, версия 1.8 и выше

Node.js - обычная установка, согласно прилагающейся инструкции, версия 12.13.0 и выше

любой браузер, например Google Chrome или Mozilla

После установки всех средств, представленных выше, необходимо включить запустить службу (Windows) или процесс (Linux/MacOS) postgres и создать базу данных со следующим названием: insurance. После создания базы данных можно приступать к запуску системы.

Инструкция пользователю по запуску приложения

Для запуска системы необходимо последовательно выполнить следующие

действия:

в папке core выполнить команду ./gradlew bootRun

проект настроен таким образом, что в базе данных автоматически создадутся все необходимые таблицы

открыть папку ui

выполнить команду npm install

выполнить команду npm build

выполнить команду npm serve

открыть, установленный браузер и ввести в адресную строку следующий адрес: http://localhost:4200/

9. Вывод

В ходе данной работы была разработана информационная система Talanto, предназначиная для организации услуг по дополнительному образованию. В процессе разработки были изучены архитектурные шаблоны, шаблоны проектирования слоев программного обеспечения. Также были пройдены следующие этапы проектирования информационной системы: выявление функциональных требований, описание бизнес-процессов, разработка вариантов использования. В результате получены полезные знания в области проектирования архитектур программного обеспечения, которые очень пригодятся в работе над реальными проектами.

С точки зрения завершенности можно оценить систему, как готовую на 90 процентов. Оставшаяся часть функциональность не несет учебной пользы и не была реализована в силу нехватки времени.

С точки зрения основных свойств распределенных систем, можно оценить системы следующим образом: система является открытой и готова к расширению, также систему можно назвать прозрачной. Особых тестов производительности не проводилось, но можно утверждать, что система точно сможет выдержать нагрузку в несколько десятков тысяч пользователей, так как ограничений для этого нет. С точки зрения удобства использования, система представлена довольно удобной и интуитивно понятной, поэтому система будет понятна сразу, даже если вы ей ещё не пользовались

Исходный код представлен на гитхабе.

Фронтенд: https://github.com/IvanovAndrey/talanto-rest-ui

Бэкенд: https://github.com/IvanovAndrey/talanto-rest-core