Разработка и интеграция клиента веб-приложения для участников и организаторов хакатонов

Проект реализуется в виде веб-приложения с доступом через десктопный ПК. Представляет собой своеобразный прототип сервиса для публикации объявлений о проведении хакатонов (главным образом по тематике разработки софта) и подачи заявок на участие в них. В ходе проектирования выделены 4 категории пользователей:

- Администратор (Administrator);

- Организатор (Organiser);

- Тренер (Trainer);

- Обычный пользователь (NormalUser, наподобие посетителя сайта).

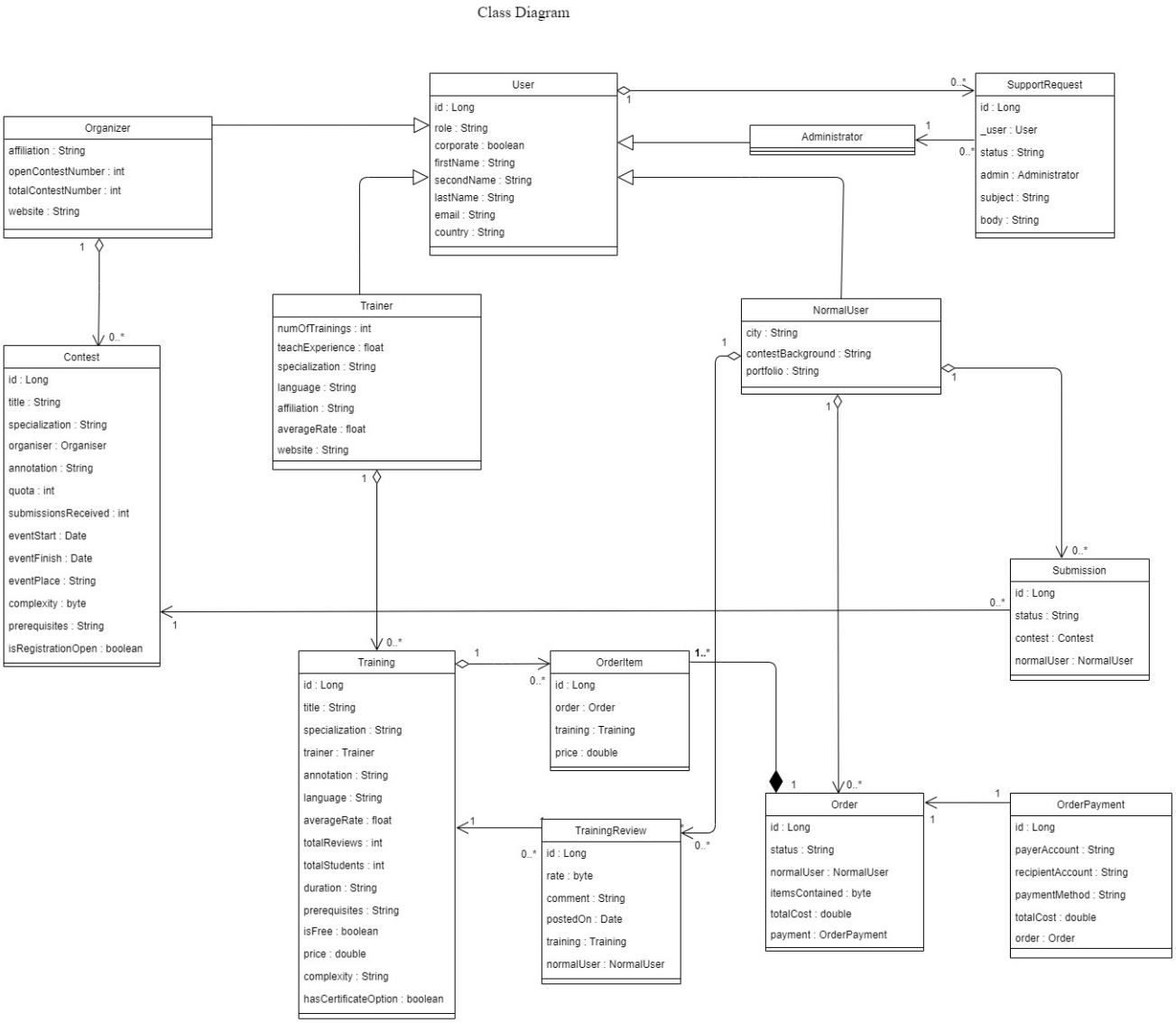
Для текущего задания имеют значение три последние категории.

1) **Организатор** через свой аккаунт публикует информацию о хакатонах определённой тематики (в коде класс хакатонов введён как Contest) и принимает заявки на участие от обычных пользователей;
2) **Тренер** через свой аккаунт размещает видеотренинги (представим, что тренинг – это некоторый курс наподобие Udemy, Coursera и др.) для покупки обычными пользователями;
3) **Обычный пользователь** в своём аккаунте просматривает объявления о хакатонах (Contest) после размещения Организатором, подаёт на них заявки (Submission) и приобретает тренинги (Training), предлагаемые Тренером.

Структура Java-пакетов

entity	классы сущностей (за исключением конкретных категорий пользователей)
entity.userclass	классы сущностей, расширяющих суперкласс User
exception	исключения
repository	классы слоя доступа к данным (DAO)
resource	классы-контроллеры
service	классы слоя сервисов
serviceimpl	имплементации сервисных классов

Диаграмма классов (также содержится в отдельном файле)



Administrator, Trainer, Organiser, NormalUser – подклассы, расширяющие **User**.

OrderItem – отдельная позиция (строка) в заказе Order. Представляет собой тренинг, выбранный обычным пользователем для покупки (один и тот же тренинг может быть указан в множестве строк, входящих в состав заказа, каждая строка соотносится с одним тренингом).

OrderPayment – оплата заказа, образующая с сущностью Order связь один-к-одному.

TrainingReview – отзыв обычного пользователя на тренинг, эта сущность не является значимой для текущего задания.

Определения свойств экземпляров классов

User

id	идентификатор пользователя
role	принадлежность к одной из 4-х категорий пользователей: ADMIN (Administrator), ORGANISER (Organiser), TRAINER (Trainer), NUSER (NormalUser)
corporate	true, если пользователь представляет организацию
firstName	имя

secondName	отчество
lastName	фамилия
email	электронная почта
country	страна, указанная пользователем при регистрации

Administrator

Не содержит специфических атрибутов

Organiser

affiliation	информация о вовлечённости оргнизатора в работу внешней организации
openContestNumber	количество хакатонов организатора, по которым ведётся регистрация участников (т.е. было объявлено о начале приёма заявок и дедлайн до сих пор не наступил)
totalContestNumber	общее число хакатонов, когда-либо объявлявшихся данным организатором
website	сайт организатора

Trainer

numOfTrainings	общее количество тренингов, предлагаемых данным тренером
teachExperience	продолжительность работы преподавателем
specialization	тематика (область) тренингов, предлагаемых тренером
language	основной язык, используемый тренером
affiliation	информация о вовлечённости тренера в работу внешней организации
averageRate	средняя оценка тренингов данного тренера
website	электронная почта

NormalUser

city	город, указанный пользователем при регистрации
contestBackground	информация об участии в хакатонах
portfolio	информация о пользовательских проектах по тематикам / в рамках хакатонов

Contest

id	идентификатор хакатона
title	название хакатона
specialization	тематика хакатона
organiser	организатор (многие-к-одному)
annotation	краткое описание, представление хакатона
quota	предел числа участников (устанавливается организатором при публикации объявления о хакатоне)
submissionsReceived	фактическое число заявок (увеличивается при передаче в обработку новых заявок)
eventStart	дата начала хакатона
eventFinish	дата окончания
eventPlace	место проведения

complexity	уровень сложности заданий
prerequisites	описание предварительных требований к участникам
isRegistrationOpen	true, если регистрация проводится в текущее время (дедлайн не наступил, организатор принимает заявки)

Training

id	идентификатор тренинга
title	название тренинга
specialization	тематика тренинга
trainer	тренер (многие-к-одному)
annotation	краткое описание, представление тренинга автором
language	основной язык тренинга
averageRate	средняя оценка тренинга обычными пользователями
totalReviews	общее число отзывов о тренинге
totalStudents	общее число купивших тренинг пользователей
duration	длительность тренинга (в виде значения String)
complexity	уровень сложности заданий
prerequisites	описание предварительных требований к участникам
isFree	true, если тренинг доступен бесплатно
price	стоимость открытия доступа к тренингу
hasCertificateOption	true, если по итогам тренинга предусмотрена выдача сертификата

Order

id	идентификатор заказа
status	статус заказа (выберем значение «ACCEPTED» в момент создания заказа и оставим без изменений)
normalUser	обычный пользователь, создавший заказ (многие-к-одному)
itemsContained	количество позиций (тренингов) в структуре заказа
totalCost	сумма к оплате
payment	оплата заказа (один-к-одному)

OrderItem

id	идентификатор строки заказа
order	заказ, включающий строку (многие-к-одному)
training	тренинг, представленный строкой заказа для совершения покупки (многие-к-одному)
price	стоимость открытия доступа к тренингу

OrderPayment

id	идентификатор процесса оплаты
payerAccount	расчётный счёт покупателя тренинга (обычного пользователя)
recipientAccount	расчётный счёт получателя
paymentMethod	признак способа оплаты (варианты: «Card», «Bank Account», «PayPal», «Samsung Pay» и др.)
totalCost	сумма к оплате
order	оплачиваемый заказ (один-к-одному)

TrainingReview

id	идентификатор отзыва о тренинге
rate	оценка тренинга автором отзыва (обычным пользователем)
comment	комментарий к поставленной оценке
postedOn	время и дата публикации отзыва
training	тренинг, на который составлен отзыв
normalUser	обычный пользователь, добавивший отзыв (многие-к-одному)

SupportRequest

id	идентификатор запроса в поддержку
_user	пользователь, направивший запрос (многие-к-одному)
status	текущий статус запроса (варианты: Received, Accepted, Handled, Managed, Closed)
admin	администратор, выполняющий обработку запроса (многие-к-одному)
subject	тема (заголовок) запроса
body	основное содержимое запроса

REST API

Использует набор базовых методов JPA/Hibernate для обмена данными с БД:

- save
- findById
- findAll
- deleteById

Для каждого класса-сущности созданы соответствующие типу данных сервисный и имплементирующий классы. Названия методов сервисных классов определяют логику выполняемых на их основе операций:

List<User> findUsersWhichCorporate() – получить список пользователей, зарегистрированных в качестве корпоративных

List<Order> findOrdersByTotalCost(double lowerBound, double upperBound) – получить список заказов, сумма которых находится в диапазоне значений

List<Contest> findContestsBySpecialization(String contestSpec) – получить список хакатонов определённой тематики

и т.д. В случае затруднений с пониманием логики тех или иных методов – уточним через мессенджер.

Суть задания

1. Используя готовые методы REST API и DAO, создать клиентов приложения для двух категорий пользователей – обычного пользователя и организатора – с обеспечением беспрепятственного обмена данными между клиентами и сервером.

Клиент должен предоставлять сервис регистрации/аутентификации пользователя. Перед началом регистрации пользователь должен выбрать свою категорию из трёх предлагаемых (Организатор, Тренер, Обычный пользователь). Соответственно, серверу необходимо динамически определить выбранную категорию и предоставить клиенту такую форму регистрации, которая согласуется с этой категорией (поскольку для каждой категории определён специфический набор атрибутов). Например пользователь выбрал «Зарегистрироваться как обычный пользователь» - в этом случае перед ним отображается форма регистрации с атрибутами обычного пользователя для передачи сведений в БД через поля класса *NormalUser*. Выбрал «Зарегистрироваться как организатор» - появляется другая форма регистрации, совместимая с полями в классе *Organiser* и т.д. Т.е. процесс регистрации един для всех категорий, но конкретные формы сбора данных зависят от заданной пользователем категории.

В форме регистрации для каждой категории должны быть представлены все поля, объявленные в соответствующем классе (кроме id и роли, указанной ранее). Таким образом, форма регистрации обычного пользователя будет включать:

- отметку о корпоративной принадлежности (corporate);

- имя;

- отчество;

- фамилию;

- мейл;

- страну;

- город;

- сведения об участии в хакатонах (*contestBackground)*;

- сведения о релевантных для участия в хакатоне проектах (*portfolio*).

По завершении регистрации пользователь переводится на страницу аутентификации для ввода логина/пароля. В случае ошибки ему предлагается предпринять следующую попытку (кол-во попыток не ограничено). В случае успеха пользователь получает доступ в свой аккаунт.

Интерфейс клиента для обычного пользователя

GUI в аккаунте обычного пользователя должен включать 5 функциональных областей:

1) область профиля пользователя (содержит сведения, указанные пользователем при регистрации: имя, отчество, фамилию, логин, страну, город, мейл);

2) область со списком поданных заявок (*Submission*) на участие в хакатоне (при отсутствии заявок выводится сообщение «Список заявок пуст»);

3) область создания новой заявки – кнопка «Создать заявку», при нажатии которой появляется форма регистрации (возможно, в виде модального окна) на хакатон со следующими полями:

- имя (заполняется автоматически);

- отчество (автоматически);

- фамилия (автоматически);

- тематика

Необходимо динамически получить из БД список тематик хакатонов с помощью метода

```
public Set<String> findContestSpecializations() {
    List<Contest> contestList = Lists.newArrayList(contestRepository.findAll());
    Set<String> specializations = new TreeSet<String>();

    for (Contest contest : contestList)
        specializations.add(contest.getSpecialization());

    return specializations;
}
```

в классе *ContestServiceImpl*. Т.е. пользователь делает клик по полю «Тематика» и сразу получает выпадающий список с опциями. Выбирает интересующую, после чего она отображается в самом поле;

- хакатон

Необходимо получить из БД список хакатонов выбранной пользователем тематики, используя метод

```
public List<Contest> findContestsBySpecialization(String contestSpec) {
    List<Contest> contestList = Lists.newArrayList(contestRepository.findAll());

    return contestList.stream()
            .filter(x -> x.getSpecialization().equals(contestSpec))
            .collect(Collectors.toList());
}
```

Т.е. пользователь делает клик по полю «Хакатон» и сразу получает выпадающий список с опциями для конкретной тематики, выбранной ранее. Выбирает интересующую опцию, после чего она отображается в самом поле.

Заполнив все поля заявки, пользователь нажимает кнопку «Отправить», после чего форма закрывается, а в области «Список заявок» появляется новая строка с краткой информацией о только что отправленной заявке (её ID, тематика, наименование хакатона). Отправленная заявка попадает в БД (с отображением через класс *Submission*) и впоследствии должна отображаться в аккаунте организатора хакатона.

4) область списка тренингов, заказанных пользователем

Содержит перечень кратких сведений о тренингах, приобретённых конкретным пользователем. При отсутствии у пользователя заказов выводится сообщение «Список тренингов пуст»;

5) область создания заказа

Содержит кнопку «Приобрести тренинг», при нажатии которой появляется электронная форма заказа (возможно, внутри модального окна) со следующими полями:

- имя (заполняется автоматически);

- отчество (автоматически);

- фамилия (автоматически);

- тематика тренинга

Необходимо получить список тематик тренингов, записи о которых хранятся в БД, по аналогии с выводом информации при составлении заявки на хакатон. Для этой цели реализуется метод

```
public Set<String> findTrainingSpecializations() {
    List<Training> trainingList = Lists.newArrayList(trainingRepository.findAll());
    Set<String> specializations = new TreeSet<String>();

    for (Training training : trainingList)
        specializations.add(training.getSpecialization());

    return specializations;
}
```

класса *TrainingServiceImpl* . Полученные данные отображаются в виде выпадающего списка, из которого пользователь выбирает интересующую опцию. После совершения выбора (допускается одновременный выбор только одной опции) название тематики демонстрируется в текущем поле;

- тренинг

Алгоритм идентичен выбору хакатона при составлении заявки. Метод получения списка данных из БД:

```
@Override
public List<Training> findTrainingsBySpecialization(String spec) {
    List<Training> trainingList = Lists.newArrayList(trainingRepository.findAll());

    return trainingList.stream()
            .filter(x -> x.getSpecialization().equals(spec))
            .collect(Collectors.toList());
}
```

Название выбранного тренинга отображается в текущем поле.

- стоимость

Определяется автоматически с учётом стоимости выбранного тренинга, т.е. одновременно с выбором тренинга в поле «Стоимость» передаётся его цена.

Таким образом, в процессе заполнения формы определяются значения полей классов *OrderItem* и *Order*.

В *OrderItem*:

order – заказ, в рамках которого происходит покупка тренинга

training – выбранный пользователем тренинг

price – стоимость тренинга (и она же стоимость всего заказа)

В *Order*:

status – определяем как «Accepted»

normalUser – заказчик

itemsContained – определяем как 1;

totalCost – определяем как стоимость выбранного тренинга.

По окончании заполнения формы пользователь нажимает кнопку «Перейти к оплате» и получает доступ к следующей электронной форме. В неё вносятся данные, необходимые для определения полей класса *OrderPayment*:

- payerAccount (вводится произвольное значение)

- recipientAccount (вводится произвольное значение)

- paymentMethod (вводится «Card», «Bank Account», «PayPal», «Samsung Pay» или др.)

- totalCost (стоимость определяется ценой тренинга и подставляется автоматически);

- order (автоматически подставляется текущий заказ).

Заполнив все поля, пользователь нажимает кнопку «Оплатить». Информация об оплате передаётся в БД, в списке тренингов появляется новая строчка с ID заказа и наименованием тренинга. Электронная форма закрывается, и пользователь возвращается на главную страницу в собственном аккаунте (ту, что демонстрирует 5 перечисленных областей).

Интерфейс клиента для организатора

Процедура регистрации / аутентификации организатора аналогична предусмотренной для обычного пользователя (при этом организатору выводится специфическая по набору полей электронная форма регистрации, как и было указано выше).

В GUI клиента организатора необходимо создать 3 функциональных области:

1) область пользовательского профиля (имя, отчество, фамилия, логин, страна, организация (при регистрации указываем произвольную), сайт);
2) область списка хакатонов, объявленных организатором (отображаются ID хакатона, его название и тематика; при отсутствии мероприятий выводится сообщение «Список мероприятий пуст»)
3) область создания нового хакатона

Содержит кнопку «Создать мероприятие», при нажатии которой появляется электронная форма создания хакатона (возможно, внутри модального окна) со следующими полями (соответствуют атрибутам экземпляра класса *Contest*):

- название;

- тематика (указываем произвольную из числа близких к разработке ПО);

- организатор (подставляется автоматически);

- краткое описание;

- максимальное число участников (значение submissionReceived устанавливается в 0);

- дата начала;

- дата окончания;

- место проведения;

- уровень сложности (по 10-тибалльной шкале);

- предварительные требования к участникам;

- открытие регистрации (возможно представить в виде чек-бокса, где наличие галки определяется как true).

Заполнив форму, пользователь нажимает кнопку «Добавить мероприятие». В результате создаётся объект Contest, через который производится отображение данных в БД. В списке хакатонов появляется новая строка с ID, наименованием и тематикой хакатона.

Через данную строку (путём нажатия на неё) у организатора должна быть возможность открыть список заявок, поступивших на данное мероприятие от обычных пользователей. Т.е. организатор делает клик по строке и названием хакатона и видит список заявок с указанием ID заявки, времени поступления и логина пользователя.

Тестирование обмена данными клиентом и сервером

1) После создания хакатона организатором обычный пользователь, выбравший аналогичную этому хакатону тематику, должен увидеть опцию с ним в списке потенциальных тренингов. Если этого не происходит – ищем баг и добиваемся требуемого результата;
2) После подачи заявки обычным пользователем сведения о ней должны быть отображены в списке участников, о котором говорится в маркированном жёлтым абзаце. Если этого не происходит – ищем баг и добиваемся требуемого результата;
3) При выполнении данных требований проводим общую проверку корректности с учётом всех изложенных в этом документе требований.