Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Научно-образовательный центр «Высшая ИТ школа»

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
д-р физ.-мат. наук, профессор

_____ О.А. Змеев
подпись
«11» июня 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ "ЛИЧНЫЙ КАБИНЕТ ШКОЛЬНИКА" ДЛЯ ТГУ

по направлению подготовки 09.03.04 Программная инженерия
направленность (профиль) «Программная инженерия»

Карагеоргий Алексей Николаевич

Руководитель ВКР
Канд. физ.-мат. наук, доцент кафедры
кафедра системного анализа и
математического моделирования, ТГУ

_____Ж.Н. Зенкова
подпись
«10» июня 2021 г.

Консультант
Начальник управления цифровых
решений ТГУ
_____Д.А. Соколов
подпись
«08» 06 2021 г.

Автор работы
студент группы № 971704

_____ А.Н. Карагеоргий
подпись
«08» июня 2021 г.

Томск – 2021

Министерство науки и высшего образования Российской Федерации.
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
НОЦ «Высшая ИТ школа»

УТВЕРЖДАЮ
руководитель ООП
д-р.физ.-мат. наук, профессор
_____ О.А.Змеев
«27» 02 2021 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы бакалавра обучающемуся
Карагеоргий Алексей Николаевич
(Ф.И.О. обучающегося)

по направлению подготовки Программная инженерия, направленность «Программная инженерия»

1. Тема выпускной квалификационной работы бакалавра
Разработка веб-приложения "Личный кабинет школьника" для ТГУ

2. Срок сдачи обучающимся выполненной выпускной квалификационной работы:
а) в учебный офис – «08» 06 2021 г.
б) в ГЭК – «11» 06 2021 г.

3. Исходные данные к работе:
Цель работы: разработка веб-приложения «Личный кабинет школьника».
цели и задачи ВКР, ожидаемые результаты

Задачи: Сбор и формализация требований заказчика к разрабатываемому сайту.

Проектирование сайта.

Реализация функционала веб-приложения в рамках первого цикла разработки.

Организация, по тематике которой выполняется работа

ТГУ, Управление цифровых решений

Руководитель выпускной квалификационной работы
канд. физ.-мат. наук, доцент кафедры
кафедра системного анализа и
математического моделирования, ТГУ _____ / Ж.Н. Зенкова
(должность, место работы) (подпись) (И.О. Фамилия)

Консультант выпускной квалификационной работы

начальник управления цифровых
решений ТГУ _____ / Д.А. Соколов
(должность, место работы) (подпись) (И.О. Фамилия)

Задание принял к исполнению
«27» 02 2021 г _____ / А. Н. Карагеоргий
(дата) (подпись) (И.О. Фамилия)

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ 5
ВВЕДЕНИЕ 6
1. Аналитика и проектирование 7
1.1 Анализ требований. Функциональные требования 7
1.1.1 Функциональные возможности неавторизованного пользователя. 7
1.1.2 Функциональные возможности авторизованного пользователя. 7
1.1.3 Функциональные возможности администратора. 8
1.2 Анализ требований. Нефункциональные требования 8
1.3 Формализация требований 8
1.4 Модель предметной области 20
2. Особенности используемых инструментов 24
2.1 Особенности архитектуры 24
2.1.1 Маршрутизация и работы с запросами 24
2.1.2 Посредник 25
2.1.3 Контролер в Laravel 27
2.1.4 Модели и взаимодействие с базой данных 27
2.1.5 Представление в Laravel 29
2.1.6 Сервис-провайдер (поставщик услуг) 32
2.1.7 Жизненный цикл запроса 32
2.2 Библиотеки и плагины 34
2.2.1 Библиотека для создания QR-кодов (SimpleQRCode) 34
2.2.2 Библиотека для редактирования информационных страниц сайта (TinyMCE Editor) 35
3. Реализация 37
3.1 Создать мероприятие 38
3.2 Сгенерировать коды для участников 41
3.3 Подтвердить посещение мероприятия 43

3.4 Создать предмет магазина 46
3.5 Приобрести предмет магазина 49
ЗАКЛЮЧЕНИЕ 52
СПИСОК ИСПОЛЬЗОВАНОЙ ЛИТЕРАТУРЫ 53

СПИСОК ИЛЛЮСТРАЦИЙ

Рисунок 1- Диаграмма вариантов использования Администратора 9
Рисунок 2 - Диаграмма вариантов использования Школьника 14
Рисунок 3- Диаграмма вариантов использования Пользователя 19
Рисунок 4 – Диаграмма классов модели Предметной Области 20
Рисунок 5 - Диаграмма классов Личного кабинета школьника, информация о пользователе. 21
Рисунок 6 - Диаграмма классов приложения «Личный кабинет школьника», взаимодействие Пользователя и Мероприятия 22
Рисунок 7- Диаграмма классов приложения «Личный кабинет школьника», магазин предметов 23
Рисунок 8 - Схема MVC паттерна в Laravel 24
Рисунок 9 – Жизненный цикл запроса/ответа 25
Рисунок 10 - Схема взаимодействия Eloquent ORM 28
Рисунок 11 – Функционал blade-шаблонов 31
Рисунок 12 – Диаграмма состояний запроса в Laravel 33
Рисунок 13 - Текстовый редактор TinyMCE 35
Рисунок 14 – Итоговая диаграмма классов веб-приложения 37
Рисунок 15 - Диаграмма активности варианта использования "Создать мероприятие" 39
Рисунок 16 - Форма создания мероприятия 40
Рисунок 17 – Страница мероприятия 41
Рисунок 18 - Диаграмма последовательности Описание генерации кодов для участников 42
Рисунок 19 – Форма генерации кодов 43
Рисунок 20 – QR-код из файла кодов мероприятия 43

Рисунок 21 - Форма ввода кода мероприятия 44

Рисунок 22 - Диаграмма последовательности Подтверждение посещения мероприятия 45

Рисунок 23 – Диаграмма активности варианта использования «Создать предмет магазина» 47

Рисунок 24 – Форма создания товара 48

Рисунок 25 – Страница магазина 49

Рисунок 26 – Диаграмма последовательности «Приобрести предмет магазина» 50

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

- Фреймворк (framework) – это программное обеспечение для создания продукта, состоящая из различных компонентов, которые упрощают разработку.
- Миграция – обновление структуры базы данных от одной версии к другой.
- Запрос (реквест, request) – это сообщение, которое посылает клиент на сервер, с целью получения ответа.
- QR-code – это двухмерный штрих код (бар-код), предоставляющий информацию для быстрого её распознавания с помощью камеры на мобильном телефоне.
- Паттерн – шаблон проектирования, который является решением часто возникающей проблемы, с точки зрения архитектуры программного обеспечения.
- Плагин – это независимый программный блок, который встраивается в приложение и расширяет его возможности.
- Маршрутизация (роутинг) – это процесс определения пути следования запроса. Используется для приема запроса с данными и передачи его на следующий этап обработки.

ВВЕДЕНИЕ

В современных реалиях для университета очень важно иметь большую базу абитуриентов, которые в будущем создадут основу для формирования и отбора студентов в университет. Ради привлечения школьников проводятся различные мероприятия, в том числе направленные на агитацию к поступлению в ВУЗ, и, порой, информация о проводимых событиях просто не доходит до абитуриентов.

Исходя из вышесказанного, принято решение, создать ресурс, в котором можно отследить все мероприятия, проводимые в рамках университета, для этого и была разработана идея по созданию сайта «Личный Кабинет Школьника».

Это веб-приложение станет информационным ресурсом для школьников, в котором представлен перечень мероприятий и информация о них, что увеличит посещаемость и привлечет новую аудиторию. Школьники получат больше информации об университете. Предполагается создание системы поощрений, в которой за посещение школьники получают баллы. На сайте предполагается раздел магазина, где можно приобрести товары с атрибутикой университета. Каждый товар обладает стоимостью, выраженной в баллах, которые накапливают пользователи.

Университет со своей стороны, привлекает предполагаемых, будущих абитуриентов, используя полученную информацию, он может выстраивать политику приемной кампании на основе данных о посещении и активности пользователей. Также это упростит примерную оценку количества посетителей мероприятий, при подготовке к нему, так как пользователь имеет два этапа отметки мероприятия. На первом этапе, пользователь регистрируется на мероприятие, это значит, что он собирается прийти на него. На втором, пользователь вводит код подтверждения посещения мероприятия, тем самым отмечает, что он побывал на нем.

В результате, целью данной работы является разработка веб-приложения «Личный Кабинет Школьника».

Проект реализуется с использованием процесса итерационной разработки. В этой работе будут рассмотрены следующие задачи:

- Сбор и формализация требований заказчика к разрабатываемому сайту.
- Проектирование сайта
- Реализация основного функционала веб-приложения в рамках первого цикла разработки.

1. Аналитика и проектирование

На первом этапе был произведен сбор и формализация пожеланий заказчика. В результате зафиксированы возможности приложения в виде сформированных требований. Требования разделены на функциональные, которые отображают доступный функционал приложения, и не функциональные, которые отображает визуальные и технические особенности. Полученные требования представлены ниже:

1.1 Анализ требований. Функциональные требования

В разрабатываемой системе принято решение разделить пользователей на роли, что позволит управлять доступом к возможностям приложения. В итоге выделены 3 роли: «администратор», «авторизованный» и «неавторизованный» пользователи. Требования к функциональным возможностям системы разделены в соответствии ролям.

1.1.1 Функциональные возможности неавторизованного пользователя.

Возможности неавторизованного пользователя являются общими и доступны пользователям с другими ролями.

- Возможность авторизации и регистрации на сайте
- Возможность просмотра предстоящих мероприятий и информации о них
- Возможность поиска мероприятия
- Возможность задать вопрос приемной комиссии

1.1.2 Функциональные возможности авторизованного пользователя.

Авторизованный пользователь, определяется как абитуриент (школьник) и приложение предоставляет ему следующий функционал:

- Возможность просмотра и редактирования информации о пользователе в профиле
- Возможность просмотра списка посещенных мероприятий
- Возможность просмотра магазина предметов
- Возможность приобретения предметов в магазине

- Возможность просмотра наград (приобретенных предметов)
- Возможность зарегистрироваться на мероприятие
- Возможность подтвердить посещение мероприятия

1.1.3 Функциональные возможности администратора.

Для администратора сайт предоставляет следующий функционал:

- Возможность создания, редактирования и удаления мероприятия.
- Возможность создания, редактирования и удаления предметов магазина
- Возможность сгенерировать QR-коды подтверждения для участников
- Возможность выгрузить данные об участниках мероприятия
- Возможность редактировать содержимое информационных страниц
- Возможность управления справочниками (категории товаров и мероприятий, роли, школы)

1.2 Анализ требований. Нефункциональные требования

- Среда разработки – Laravel
- Синхронизация вопросов к приёмной комиссии с сайтом абитуриента (abiturient.tsu.ru)
- Вывод мероприятий в формате календаря

1.3 Формализация требований

После анализа требований заказчика, вся полученная информация была формализована и приведена к сценариям и диаграммам вариантов использования.

Как уже было сказано, в системе все пользователи разделены на три роли: «администратор», «авторизованный пользователь» (школьник), «неавторизованный пользователь».

Далее рассмотрим варианты использования «администратора».

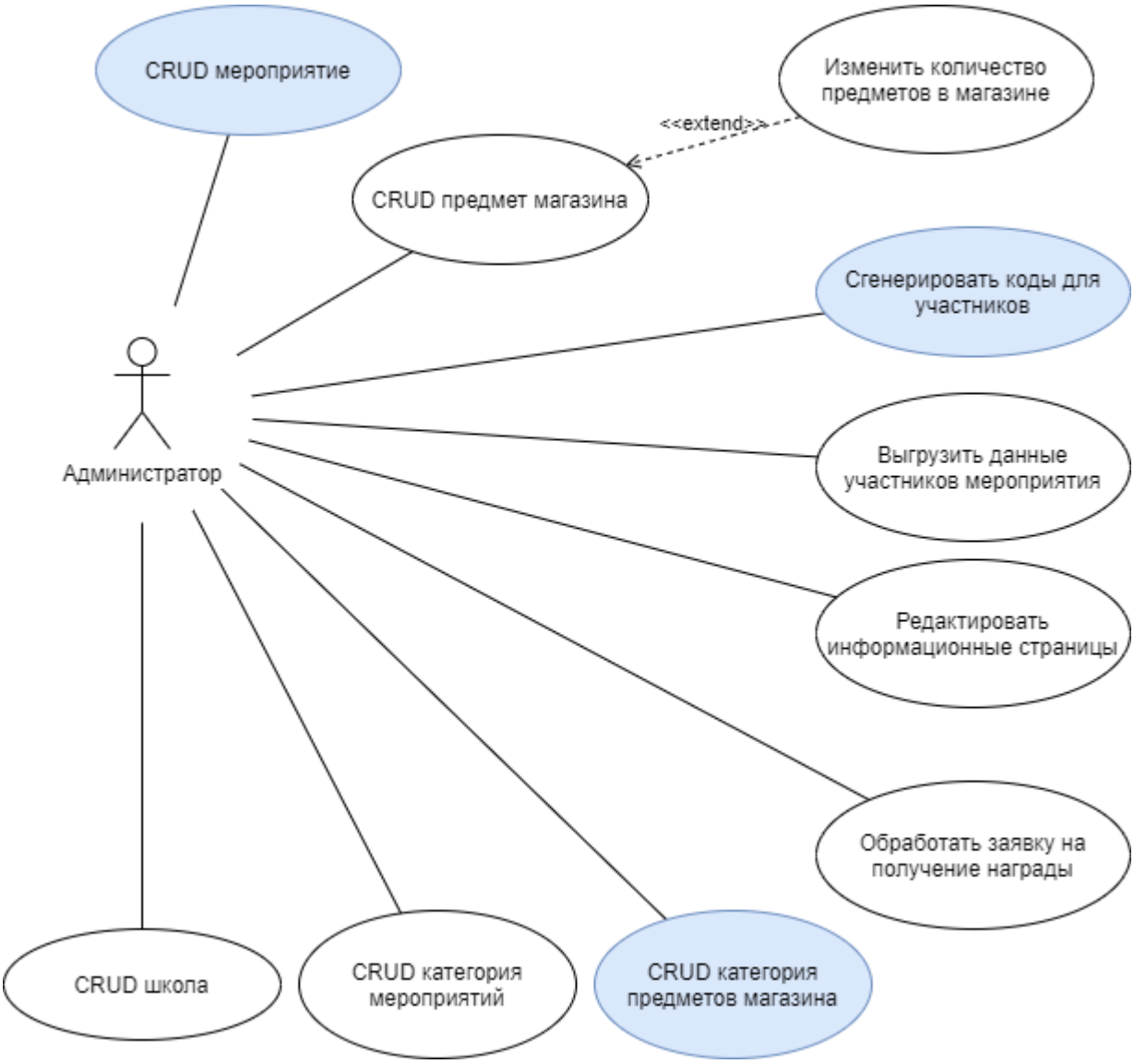


Рисунок 1- Диаграмма вариантов использования Администратора

На диаграмме выше, представлены все доступные администратору варианты использования, среди них цветом выделены архитектурно-значимые. Основная задача администратора заключается в управлении сущностями приложения. Система разделена несколько частей, в каждой из которых будут выделены архитектурно значимые варианты использования. Главные задачи приложения заключаются в предоставлении информации о мероприятии и работе с регистраций и отметкой посещения их. Поэтому рассмотрим более подробно варианты использования «Создать мероприятие», «Сгенерировать коды для участников».

Помимо мероприятий, приложение включает магазин предметов. К архитектурно значимым вариантам использования этой части приложения, относится «Создать предмет магазина».

Таблица 1 Описание и сценарий варианта использования "Создать мероприятие"

ВИ	**Создать мероприятие**
Цель	Создать мероприятие
Описание	Создает страницу мероприятия и отображает в списке мероприятий
Актер	Администратор
Предусловие	Пользователь открыл страницу мероприятий.
Сценарий	1. Пользователь переходит на страницу создания мероприятия. 2. Система отображает форму создания мероприятия. 3. Пользователь заполняет форму. 4. Система проверяет полученные данные. 5. Система сохраняет информацию о мероприятии в базе данных
Расширение	4.1 Полученные данные введены не верно 4.1.1 Система отображает форму создания мероприятия с заполненными полями и отмечает те, в которых допущена ошибка 4.1.2 Пользователь исправляет данные и отправляет форму.

Результат	Создано новое мероприятие, добавлена карточка мероприятия в список.

Таблица 2- Описание и сценарий варианта использования “ Сгенерировать коды для участников "

ВИ	**Сгенерировать коды для участников**
Цель	Сгенерировать QR-коды для участников
Описание	Пользователь создает новые коды мероприятия. Коды генерируются в формате QR-кода.
Актер	Администратор
Предусловие	Пользователь открыл страницу мероприятия.
Сценарий	1. Пользователь переходит открывает форму генерации кодов мероприятия. 2. Система отображает форму генерации кодов 3. Пользователь заполняет форму и отправляет её 4. Система проверяет полученные данные 5. Система создает коды для активации мероприятия, в соответствии с данными из формы. 6. Система переводит коды мероприятия в формат QR-кода. 7. Система создает pdf-файл и добавляет в него QR-коды. 8. Система отправляет пользователю, созданный pdf-файл. 9. Пользователь получает файл с QR-кодами для участников мероприятия.

Расширение	4.1 Полученные данные введены не верно 4.1.1 Система отображает оповещение об ошибке 4.1.2 Пользователь вводит исправленные данные
Результат	Пользователь получает pdf-файл с QR-кодами для участников мероприятия.

Таблица 3 - Описание и сценарий варианта использования "Создать предмет магазина"

ВИ	**Создать предмет магазина**
Цель	Создать предмет магазина
Описание	Администратор создает новый предмет магазина, который могут приобрести пользователи.
Актер	Администратор
Предусловие	Пользователь открыл страницу магазина.
Сценарий	1. Пользователь переходит на страницу создания предмета магазина. 2. Система отображает форму создания предмета магазина. 3. Пользователь заполняет форму. 4. Система проверяет полученные данные. 5. Система сохраняет информацию о предмете магазина в базе данных. Система добавляет запись о кол-ве предметов магазина доступных к покупке.

Расширение	4.1 Полученные данные введены не верно 4.1.1 Система отображает форму создания предмета магазина с заполненными полями и отмечает те, в которых допущена ошибка 4.1.2 Пользователь исправляет данные и отправляет форму.
Результат	Создан новый предмет магазина, добавлена информация о его количестве.

Школьник является еще одним важным пользователем приложения. Рассмотрим его варианты использования системы.

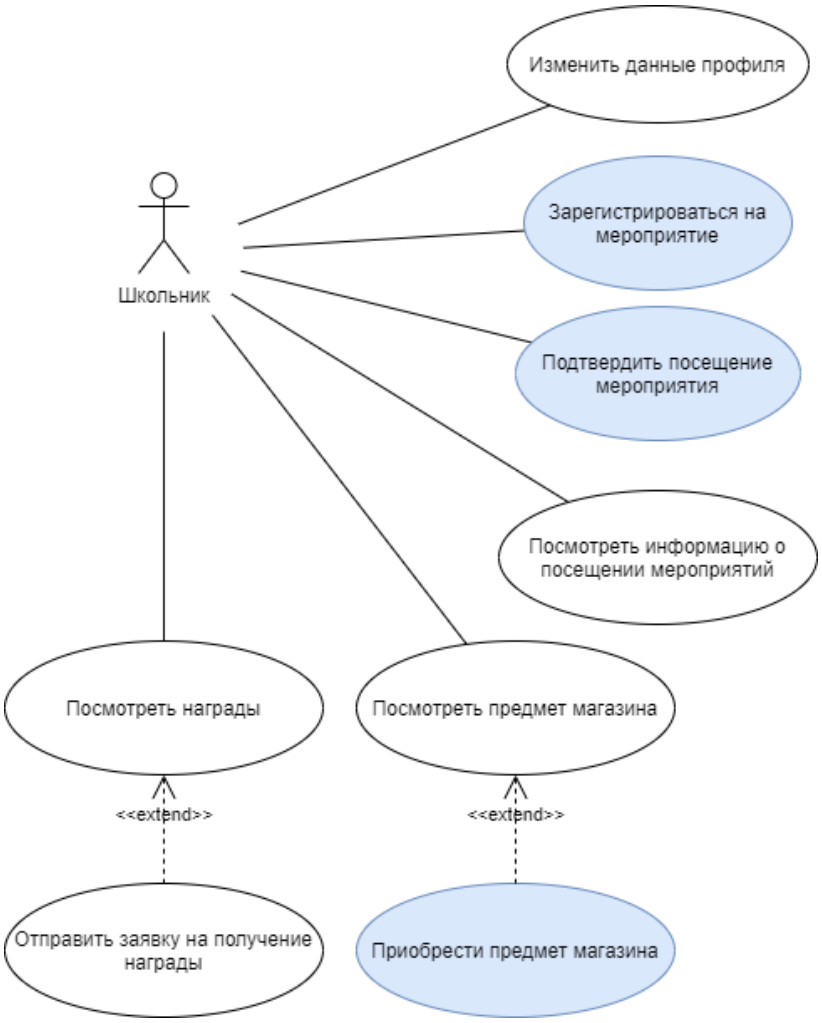


Рисунок 2 - Диаграмма вариантов использования Школьника

На диаграмме отображены варианты использования системы для пользователя с ролью «Школьник». Каждый школьник может регистрироваться на мероприятия, а также ему доступен страница профиля. После того как мероприятие пройдет, школьник сможет подтвердить его посещение. Подтверждение посещения мероприятия происходит с помощью кода мероприятия. Школьник получает от организатора QR-код, в котором записан код мероприятия, и вводит его в форму активации в профиле. После активации кода пользователь получает баллы. Личный профиль пользователя содержит персональные данные и информацию о мероприятиях, которые он посетил или на которые только зарегистрировался.

Полученные за посещения мероприятий баллы накапливаются, и школьник может с их помощью приобрести предмет в магазине. Купленный предмет будет отображаться в наградах. После того как предмет появится на

странице наград, школьник заполняет заявку на получение награды. Заявка представляет из себя форму с информацией о способе выдачи и данных, необходимых для отправки по почте.

Среди представленных вариантов использования были выделены архитектурно значимые, они отмечены цветом на изображении: «Зарегистрироваться на мероприятие», «Подтвердить посещение мероприятия», «Приобрести предмет магазина», «Отправить заявку на получение награды». Для выделенных архитектурно значимых вариантов использования были написаны сценарии и описания вариантов использования, представленные ниже.

Таблица 4 - Описание и сценарий варианта использования "Зарегистрироваться на мероприятие"

ВИ	**Зарегистрироваться на мероприятие**
Цель	Зарегистрироваться на мероприятие
Описание	Пользователь регистрируется на мероприятие.
Актер	Школьник
Предусловие	Пользователь находится на странице мероприятий
Сценарий	1. Пользователь выбирает мероприятие и открывает его страницу. 2. Система отображает страницу мероприятия. 3. Пользователь нажимает кнопку регистрации на мероприятие. 4. Система проверяет статус регистрации пользователя на мероприятие. 5. Система регистрирует пользователя на мероприятие. 6. Система отображает сообщение об успешной регистрации.

Расширение	4.1 Пользователь уже зарегистрирован 4.1.1 Система отображает сообщение, что пользователь уже зарегистрирован
Результат	Пользователь регистрируется на мероприятие. Мероприятие появляется в профиле пользователя, в разделе “Мои мероприятия”

Таблица 5 - Описание и сценарий варианта использования " Подтвердить посещение мероприятия"

ВИ	**Подтвердить посещение мероприятия**
Цель	Подтвердить посещение мероприятия
Описание	Пользователь активирует код и подтверждает посещение мероприятия. За активацию он получает баллы.
Актер	Школьник
Предусловие	Пользователь посетил мероприятие и получил QR-код. Он просканировал QR-код и считал из него код мероприятия.
Сценарий	1. Пользователь открывает страницу профиля 2. Система отображает профиль пользователя 3. Пользователь открывает форму активации кода 4. Система отображает форму активации кода 5. Пользователь вводит и отправляет код подтверждения посещения мероприятия 6. Система находит код и проверяет его статус 7. Система добавляет мероприятие в список посещенных пользователем.

	8. Система выдает баллы за мероприятие пользователю 9. Пользователь получает сообщения о получении баллов за мероприятие.
Расширение	6.1 Кода не существует. 6.1.1 Система отображает сообщения о том, что такого кода не существует. 6.1.2 Пользователь вводит исправленный код. 6.2 Код уже активирован 6.2.2 Система отображает уведомление о том, что введенный код уже активирован 6.2.3 Пользователь вводит исправленный код
Результат	Пользователь подтверждает посещение мероприятия. Он получает баллы мероприятия. Мероприятие отображается в профиле в разделе «Моя активность», с пометкой о посещении.

Таблица 6 - Описание и сценарий варианта использования "Приобрести предмет магазина"

ВИ	**Приобрести предмет магазина**
Цель	Приобрести предмет в магазине приложения.
Описание	Пользователь приобретает предмет в магазине приложения.
Актер	Школьник
Предусловие	Пользователь находится на странице профиля.

Сценарий	1. Пользователь открывает страницу магазина сайта. 2. Система отображает страницу с предметами магазина. 3. Пользователь выбирает предмет и покупает его. 4. Система проверяет количество предметов в продаже. 5. Система проверяет возможность приобретения товаров пользователем. 6. Система создает запись о покупке предмета магазина и списывает баллы с пользователя. 7. Система отображает сообщение об успешной покупке.
Результат	Пользователь приобретает предмет в магазине. Предмет отображается в профиле в разделе «Мои награды».

Таблица 7 - Описание и сценарий варианта использования "Отправить заявку на получение награды"

ВИ	**Отправить заявку на получение награды**
Цель	Отправить заявку на получение награды.
Описание	Пользователь отправляет заявку, чтобы получить приобретенный предмет магазина.
Актер	Школьник
Предусловие	Пользователь приобрел предмет в магазине. Пользователь находится на странице профиля.
Сценарий	1. Пользователь открывает раздел «Мои награды». 2. Система отображает все, когда-либо купленные, предметы магазина.

	3. Пользователь выбирает предмет и нажимает кнопку «забрать». 4. Система отображает форму для получения предмета. 5. Пользователь указывает способ получения и заполняет необходимые данные. Отправляет форму. 6. Система меняет статус предмета, отображает заявку в листе заявок на получение наград.
Результат	Пользователь отправил заявку на получение предмета. Система отображает заявку в панели администратора.

Помимо вариантов использования относящихся к конкретным ролям есть те, что являются общими для всех пользователей приложения.

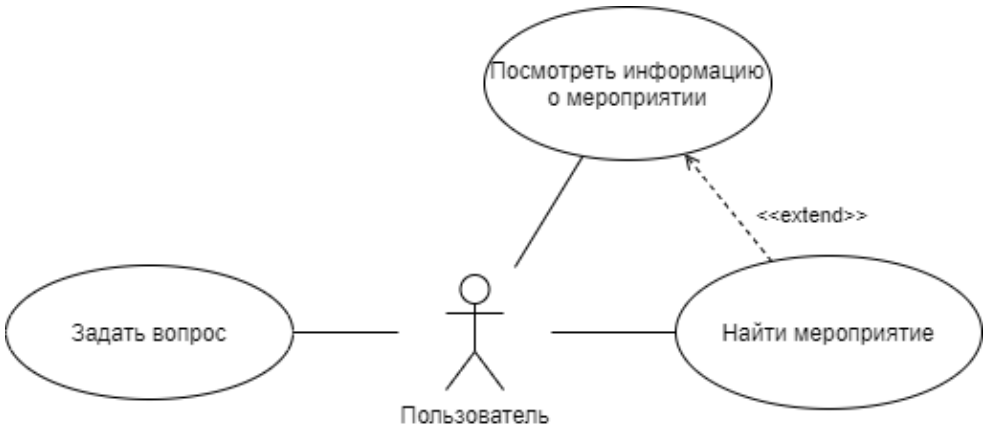


Рисунок 3- Диаграмма вариантов использования Пользователя

Варианты использования, представленные на диаграмме доступны всем пользователям. Поиск и просмотр мероприятий являются самыми важными в этой модели предметной области. Следующим этапом в разработке было проектирование модели предметной области.

1.4 Модель предметной области

В результате анализа и формализации требований, была сформирована модель предметной области.

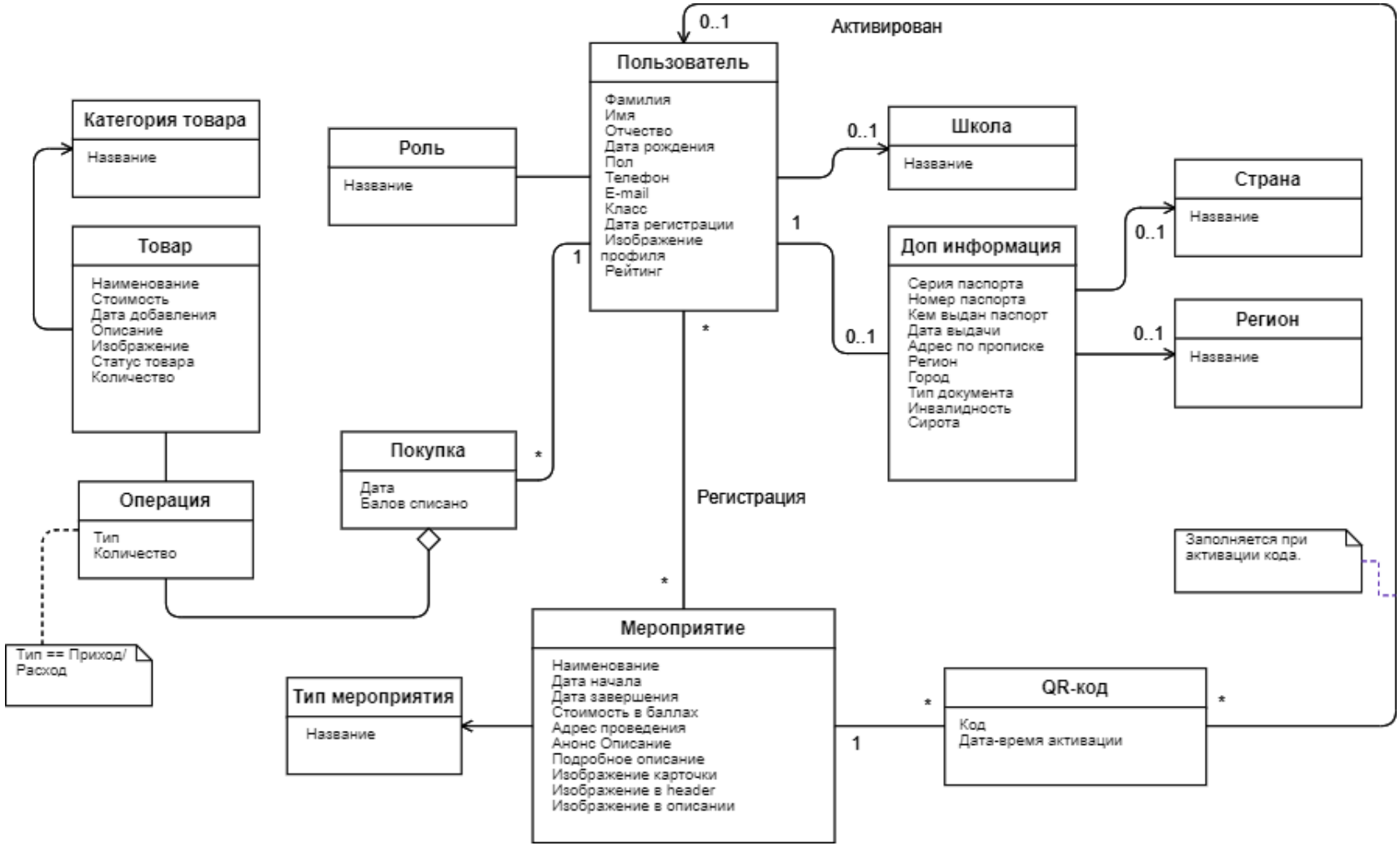


Рисунок 4 – Диаграмма классов модели Предметной Области

На диаграмме зафиксированы особенности предметной области, она включает сущности, отвечающие за различный функционал системы. Рассмотрим более подробно архитектуру и элементы приложения, разделив их по функционалу.

Рисунок 5 - Диаграмма классов Личного кабинета школьника, информация о пользователе.

Сущность «Пользователь» хранит общую информацию о человеке. Пользователь также может выбрать страну и регион проживания из списка, а также указать свою школу. При регистрации у него есть возможность указать личные персональные данные, которые сохранятся в сущности «Доп. информация». Данные используются организаторами мероприятий.

Связь «Пользователь» и «Роль» позволяет ограничивать и выдавать разрешения доступа к различным страницам приложения.

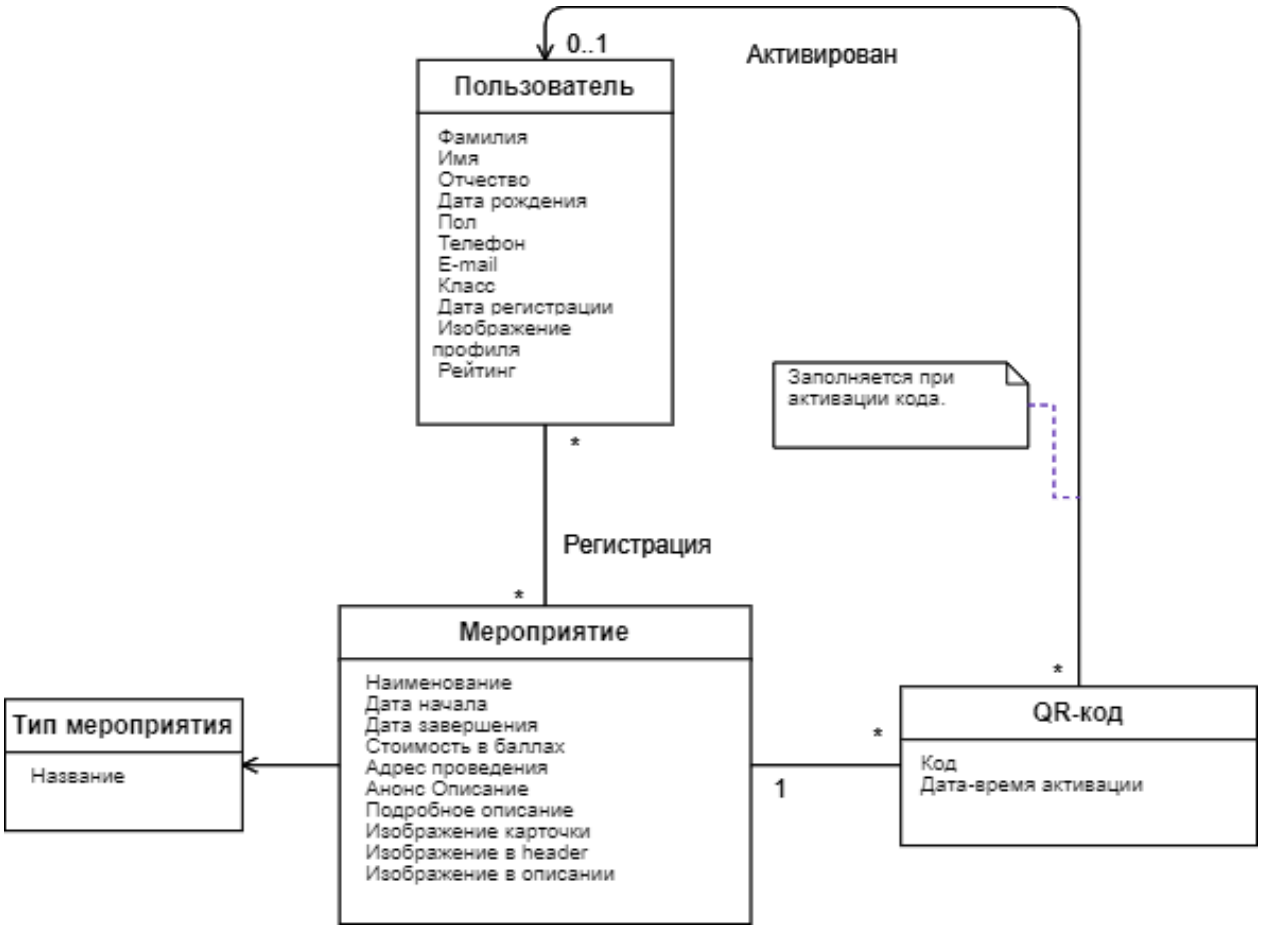


Рисунок 6 - Диаграмма классов приложения «Личный кабинет школьника», взаимодействие Пользователя и Мероприятия

На диаграмме представлено взаимодействие между пользователем и мероприятием. Рассмотрим подробнее что означает каждая связь. Первая, многие-ко-многим между объектами «Пользователь» и «Мероприятие», означает для системы, что пользователь зарегистрировался на мероприятие, эта связь используется администратором для получения списка зарегистрированных на мероприятие пользователей, а пользователь, после регистрации, увидит мероприятие в профиле, в разделе «Мои мероприятия».

Следующая связь между Мероприятием и Пользователем осуществляется через объект QR-код. В объекте QR-код, хранится код от мероприятия. К каждому мероприятию относится несколько QR-кодов. Когда пользователь введет код, система отыщет соответствующий этому коду экземпляр класса QR-код и внесет в него пользователя. С помощью этой связи отмечается пользователь, который ввел код, что подтверждает посещение данного мероприятия.

Рисунок 7- Диаграмма классов приложения «Личный кабинет школьника», магазин предметов

Следующая диаграмма описывает часть приложения, отвечающая за магазин и его функционал. «Товар» — главная сущность этой части приложения, она содержит информацию о продаваемом предмете и его количестве. Товары поделены на категории. Над товаром может производится два типа операций – приход и расход. Приход соответствует поступлению товара в магазин и увеличивает количество товара в продаже. Расход соответствует покупке товара в магазине и уменьшает количество товара. Сущность «Операция» позволяет контролировать движение товаров. При покупке товаров также задействована сущность «Покупка», она хранит общую стоимость купленных товаров и ссылки на них соответственно.

2. Особенности используемых инструментов

2.1 Особенности архитектуры

Для разработки веб-приложения было решено использовать Laravel. Это php фреймворк основанный на MVC (Model View Controller) архитектуре. Паттерн MVC является одним из самых популярных в сфере разработки. Его идея заключается в разделения функциональной части приложения на слои: модель, контроллер, представление. Такое решение уменьшает связи между компонентами системы и упрощает последующее изменение в приложении. Разделяет логику приложения, визуальное представление и информацию на отдельные части.

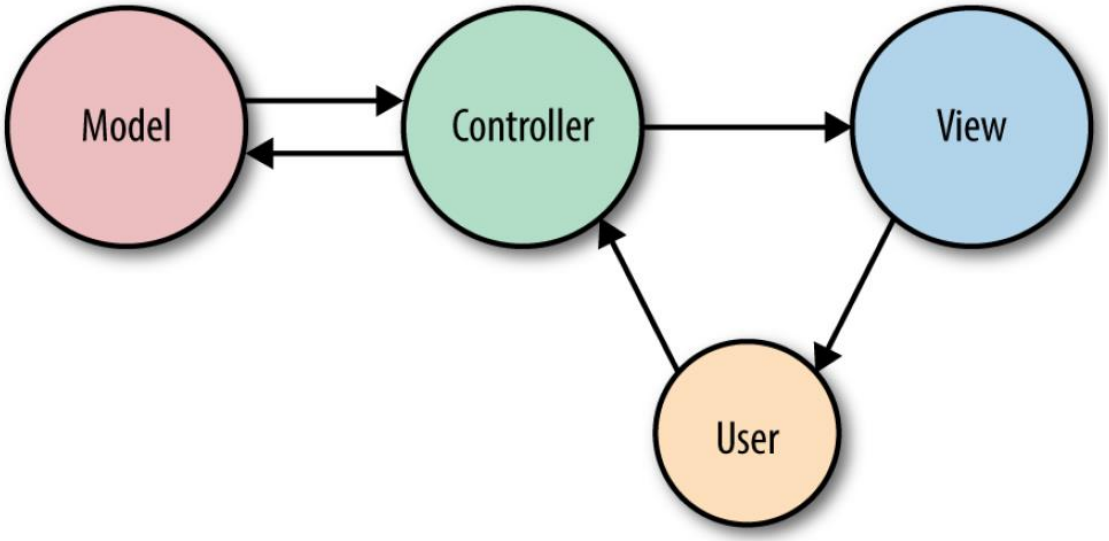


Рисунок 8 - Схема MVC паттерна в Laravel

В Laravel он реализован по схеме, отображенной выше. Пользователь сначала отправляет HTTP-запросы в контроллер. В ответ на запрос он может получить, изменить или записать данные модели. После обработки контроллер отправляет данные в представление, которое будет выведено пользователю в качестве ответа на запрос.

Далее рассмотрим компоненты фреймворка, которые отвечают за процесс обработки запросов.

2.1.1 Маршрутизация и работы с запросами

Взаимодействие с сайтом и открытие определенных станиц происходит путем обращения на различные адреса сайта. В Laravel, шаблоны адресов называются маршрутами. Все маршруты прописываются в файлах

маршрутизации в папке route. Определены два типа маршрутов *api* и *web*, они располагаются в соответствующих названию файлах. Web и Api маршруты различаются набором посредников, также у web есть возможность использования информации о сессии, клиенте и CRSF. Api – чаще всего предназначены для использования сторонними приложениями

Маршрутизатор поддерживает все типы HTTP-запросов.

Все маршруты написаны, по примеру, представленному ниже:

Листинг 1 - Пример инициализации маршрутов

```
2  Route::get('/events/{event}', 'EventsController@show');
3
4  Route::get('about', function () {
5      $info_view = DB::table('info_view')->where('url', url('about'))->first();
6      return view('About', compact( var_name: 'info_view'));
7  });
```

Существует два основных варианта определения хода выполнения запроса. Первый представлен на строке 2, когда в скобках указывается шаблон адреса и наименование контроллера и функцию, в которые запрос должен быть перенаправлен. Второй вариант на строке 4, когда непосредственно функция-обработчик запроса прописана в месте определения маршрута. Такой способ удобен, когда обработка не содержит сложной логики. К примеру, когда нам нужно перенаправить запрос на другой маршрут из текущего или сразу вывести представление.

2.1.2 Посредник

Посредник или middleware – это промежуточный элемент системы, который участвует в фильтрации и обработке запросов. Каждый middleware представляется в виде класса с одной функцией - обработчиком запроса. Посредники, могут выполнятся перед обработкой запроса или после, тем самым завершая её. Схема пути запроса показана на изображении ниже.

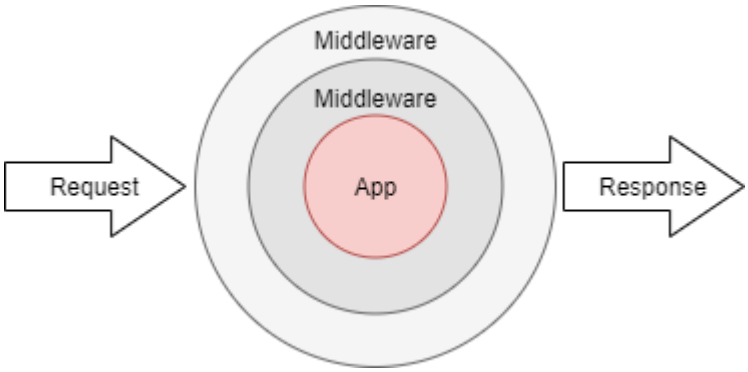


Рисунок 9 – Жизненный цикл запроса/ответа

Middleware представляют собой слои, которые образуют набор посредников. Запрос проходит через эту цепочку посредников и только потом поступает в приложение. После завершения, ответ на запрос снова проходит через посредников и отправляется пользователю.

В качестве примера, рассмотрим посредник проверки ролей (RoleMiddleware), который представлен на листинге ниже. Он отвечает за проверку роли пользователя и его доступа к текущему маршруту. Подключение посредника к маршруту происходит по примеру, представленному ниже. После маршрута в функции *middleware()* указывается название посредника (‘role’) и параметр (‘admin’) через двоеточие, при его необходимости.

Листинг 2 - Пример использования посредников для маршрутов

```
2  Route::get('events/create', 'EventsController@create')->middleware('role:admin');
3  Route::post('events', 'EventsController@store')->middleware('role:admin');
```

После того как посредник определён для маршрута, все запросы по этому маршруту будут проходить через него. Рассмотрим реализацию данного посредника на листинге ниже. При поступлении запроса, происходит проверка авторизации пользователя. Если он не авторизован, то его перенаправляют на страницу входа. Далее проверяется роль пользователя, отправившего запрос. Если роль соответствует указанной ранее в маршруте, то запрос передается дальше в обработку. В случае несовпадения роли, выдается ошибка доступа к странице.

Листинг 3 – Реализация посредника RoleMiddleware

```
7   class RoleMiddleware
8   {
9       /**
10       * Handle an incoming request.
11       *
12       * @param \Illuminate\Http\Request $request
13       * @param \Closure $next
14       * @return mixed
15       */
16      public function handle($request, Closure $next, $role)
17      {
18          if(!auth()->user()){
19              return redirect('login');
20          }
21          if (!auth()->user()->hasRole($role)) {
22              abort(404);
23          }
24          return $next($request);
25      }
26  }
```

Посредники делятся на два типа – глобальные и для определённого маршрута. Глобальные прописываются в файле-конфигурации приложения *(config/app.php)* и будут применены ко всем маршрутам. Посредники для определенного маршрута подключаются в файле маршрутов, с помощью функции *middleware ()* и указания названия посредника в параметре функции.

2.1.3 Контролер в Laravel

Контроллеры – это классы, которые включают функции обработки, относящиеся к одной модели или ресурсу приложения. Они группируют схожие функции, к примеру, CRUD одной из моделей.

Функции контроллера содержат логику обработки запроса. Она включает запросы данных моделей и обращения к сторонним классам приложения. Контроллер отвечает за преобразование и работу с данными приложения. В завершении обработки в контроллере, выполняется вызов шаблона результирующей страницы. В шаблон посылаются данные на вывод.

2.1.4 Модели и взаимодействие с базой данных

Работа с базой данных (БД) важнейшая составляющая приложения, ведь в ней хранится вся информация, которая выводится пользователям. Для

взаимодействия с БД в Laravel имеется ряд особенностей, позволяющих упростить и ускорить работу по реализации сайта.

Изменение структуры базы данных происходит с помощью миграций. Миграции – это система управления версиями для БД. С помощью миграций происходит управления полями, таблицами и связями между ними. В миграции есть описание изменений в архитектуре.

Для более легкого доступа к данным из БД в Laravel используется Eloquent ORM (Object-Relational-Mapping). Eloquent ORM – это реализация шаблона Active Record в Laravel для работы с данными.

Active Record – шаблон проектирования, который подходит для доступа к данным реляционных баз данных в объектно-ориентированном программировании. Его задача замена SQL (Structured query language) запросов на простую цепочку методов, которые при последующей обработке сформируют запрос SQL.

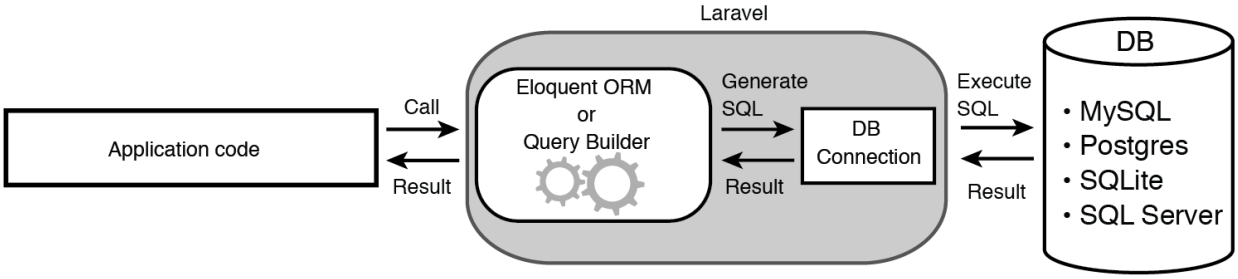


Рисунок 10 - Схема взаимодействия Eloquent ORM

На рисунке выше, отображена схема работы Eloquent ORM. Вызов осуществляется при обращении контроллера к данным Модели или написании запроса, с использованием синтаксиса Eloquent ORM. При получении вызова, Eloquent ORM генерирует необходимый SQL запрос и отправляет его к базе данных. Получив ответ, она его преобразует и возвращает системе.

Рассмотрим концепцию AR подробнее, допустим у нас существует таблица в базе данных. Для одной таблицы создается класс Active Record – модель, которая является представлением таблицы и имеет следующую концепцию:

- Каждый экземпляр класса является отражением одной записи БД
- При создании нового экземпляра класса создается новая запись в БД
- При чтении полей экземпляра, считываются значения записи БД, соответствующие классу.

- При изменении объекта, изменяется и соответствующая запись.

Для начала работы с Eloquent ORM необходимо создать модель, название модели соответствует названию таблицы в базе данных. В дальнейшем вся работа с этой таблицей может производиться с помощью неё. Перед изменением данных в БД, необходимо прописать поля таблицы в свойстве fillable, текущей модели.

Также в модели прописываются связи с другими таблицами. С их помощью можно получить данные о связанных объектах.

Листинг 4 – Класс модели User с функцией получения списка мероприятий пользователя

```
11  class User extends Authenticatable
12  {
13      use Notifiable, HasRole, HasApiTokens;
14
15      public function events()
16      {
17          return $this->belongsToMany( related: Event::class)->withTimestamps();
18      }
```

Например, при реализации связи «Многие-ко-многим» между «Пользователем» и «Мероприятием», прописывается в Модели *User* и *Event* соответственно, функции связки с *events ()* и *users ()*. Вызвав в модели *User* метод *events ()*, получим список мероприятий, связанных с пользователем.

2.1.5 Представление в Laravel

Вывод информации и представлений страниц сайта происходит с помощью view-представлений. В Laravel используется собственный шаблонизатор blade, с набором директив. Blade-шаблоны обладают широким спектром возможностей.

Основная идея blade-шаблонов — это создание гибкой среды для написания представлений. Используя их, можно структурировать и модифицировать содержимое страниц или использовать вложенное наследование, сложные условия и рекурсию.

Для упрощения создания шаблонов в Laravel существует ряд особенностей в синтаксисе, рассмотрим их более подробно. Для отображения данных Php-кода используются двойные фигурные скобки *{{$variable}}*, в них записываются данные для вывода. Также изменены управляющие конструкции:

1. Условные конструкции. Для выставления условия вывода используются выражения @if ($condition), @else, @elseif ($condition) и @endif. С их помощью можно составить любое необходимое условие для шаблонов.
2. Циклы. Для создания циклов используются выражения @for, @foreach, @while, которые работают также, как и PHP. Помимо этого, в Blade добавлено выражение @forelse – это цикл, который выполняется даже для пустого объекта. Пример его использования представлен ниже.

Листинг 5 – Пример использования @forelse в шаблоне страницы

```
19    @foreach ($users as $user)
20        <p>This is user {{ $user->id }}</p>
21    @endforeach
22    @forelse ($users as $user)
23        <li>{{ $user->name }}</li>
24    @empty
```

В данном случае цикл пройдется по всем записям из коллекции $users. Если коллекция окажется пустой, будет выполнена часть, прописанная в @empty. В примере система выведет надпись “No users”.

Все вышеуказанные выражения – это директивы. Помимо использования встроенных, можно создать собственные. Это позволяет уменьшить количество повторяющегося кода и упростить его изменение. К примеру, создать директиву проверки роли пользователя @hasrole (‘role’) и выводить различные блоки в зависимости от роли пользователя. Также есть директива проверки статуса пользователя - @guest, @auth, гость он или авторизован, соответственно.

Одной из важных особенностей Blade – это возможность наследования шаблонов, что позволяет представлениям включать в себя другие. Для этого используются следующие выражения:

1. @yield конструкция определяет раздел в шаблоне и имеет вид @yield (‘content’, ‘Home’). Первый параметр название раздела, второй стандартное значение. В этот раздел можно будет добавить контент позже.
2. @extends конструкция определяет, что представление использует другой шаблон, расширяет его. Тем самым указывает что текущее

представление устанавливает содержимое разделов. Файл-шаблон должен расширять только один другой файл. К примеру, конструкция @extends ('layouts.app') подключает шаблон *layouts/app* к текущему представлению, роль которого добавить содержимое в разделы этого шаблона.

3. @section и @endsection используются для определения содержимого раздела. Разделы, созданные в файле шаблоне функцией @yield, заполняются в дочернем представлении содержимым @section.
4. @include используется для включения, при необходимости вставить другой шаблон в текущий. Это могут быть формы или заголовки страниц, те компоненты, которые часто используемы. Их можно определить в отдельном файле и подключать данной функцией.

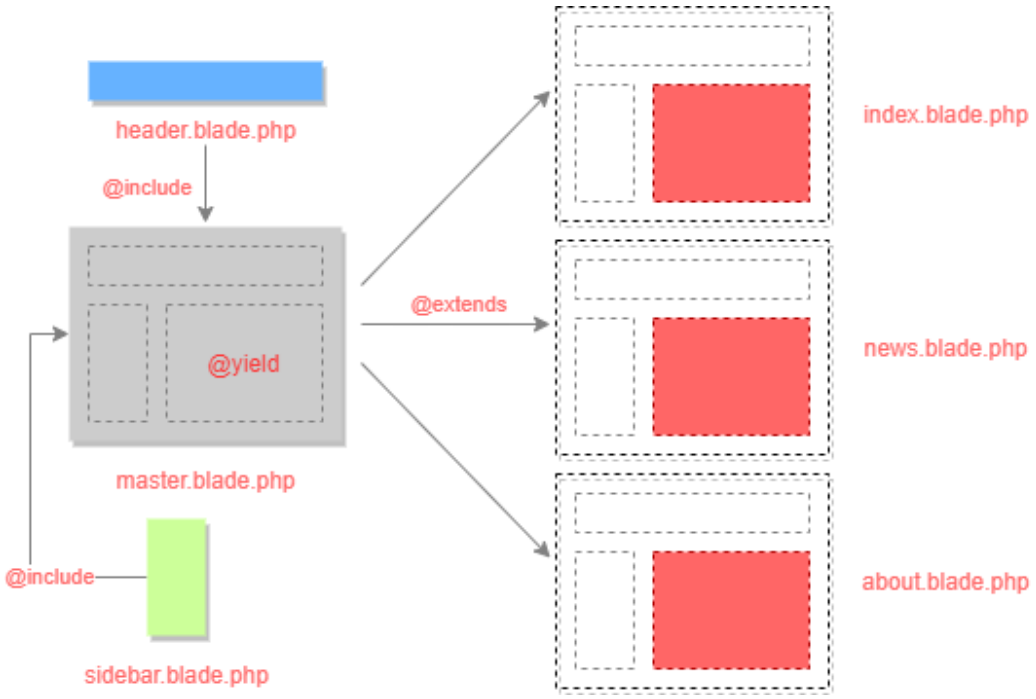


Рисунок 11 – Функционал blade-шаблонов

На изображении выше представлен пример взаимодействий blade-шаблонов. Файл *master.blade.php* является шаблоном страниц. В него включаются blade-шаблоны header и sidebar, с помощью функции @include. При генерации страницы, в место подключения будет вставлен код файла *header* и *sidebar*. @yield выделяет блок в шаблоне под контент страницы. Файлы news, index подключают шаблон функцией @extends, то есть полностью используют структуру файла *master*. Для того чтобы заполнить @yield('content'), выделенный в шаблоне, добавляет содержимое в @section('content').

Laravel blade шаблон – это удобный инструмент для создания и структурирования страниц. Он прост в использовании, но при необходимости предоставляют функционал для решения сложных задач.

2.1.6 Сервис-провайдер (поставщик услуг)

Сервис-провайдер (поставщик услуг) – это класс, содержащий логику, которую должны запускать части приложения для начальной загрузки функциональности. Сервис-провайдер представляет собой инструмент для кода начальной загрузки, который должен выполниться на этапе инициализации приложения.

Например, класс AuthServiceProvider производит начальную загрузку всех регистрационных данных, необходимых для работы системы аутентификации в Laravel.

Сервис-провайдеры имеют два метода *boot ()* и *register ()*. При запуске приложения, сначала выполняются все методы *register ()*. После того как все сервис-провайдеры зарегистрированы, вызываются методы *boot (),* для выполнения остальной части загрузки.

2.1.7 Жизненный цикл запроса

После того как были описаны элементы фреймворка в отдельности, для понимания работы приложения, далее представлено описание жизненного цикла запроса. Которое отображает взаимодействие элементов и выполнение обработки вызовов.

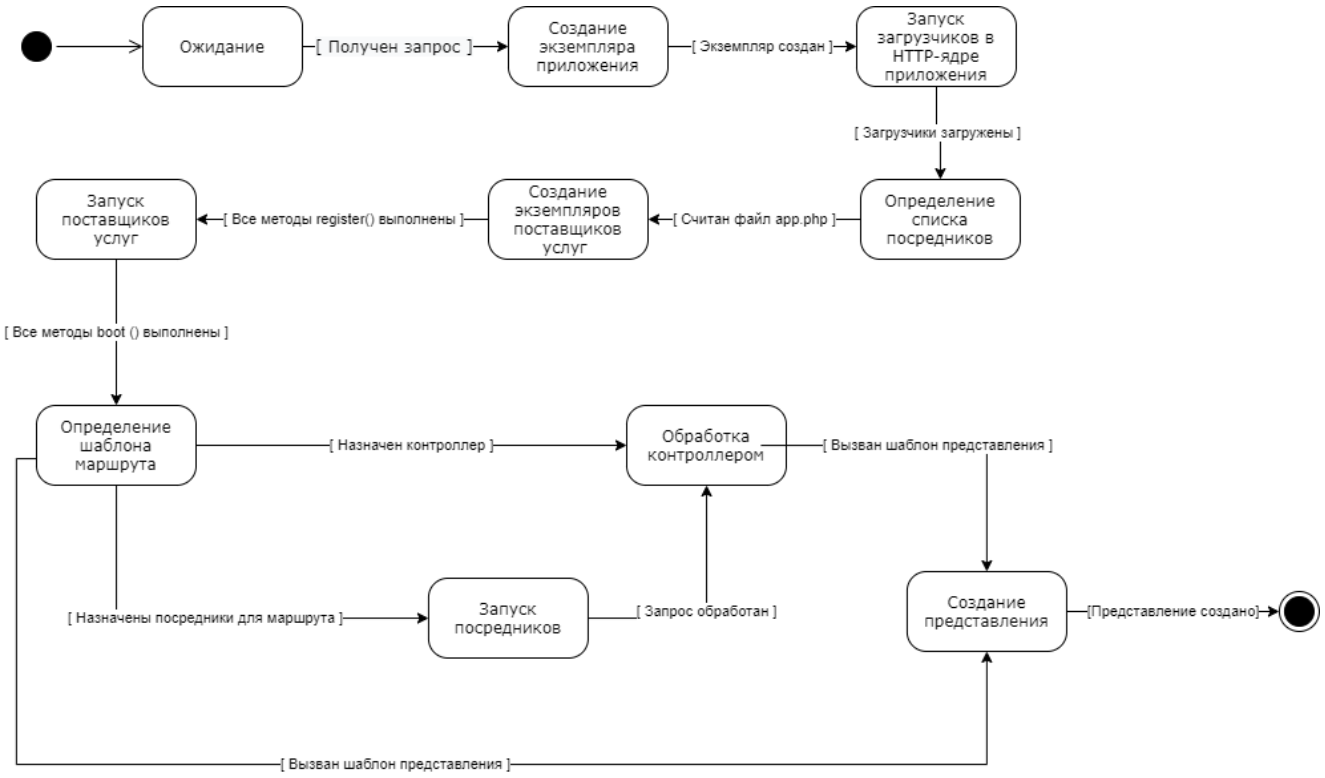


Рисунок 12 – Диаграмма состояний запроса в Laravel

На изображении выше, представлена диаграмма состояний запроса в Laravel. Когда пользователь отправляет запрос, он попадает на сервер и всегда направляется в точку входа – файл *index.php*. Он загружает файлы автоматического загрузчика. А затем создает экземпляр Laravel приложения из файла *bootstrap/app.php,* который и будет обрабатывать поступивший запрос.

Следующий шаг происходит в части ядра приложения. Входящий запрос отправляется в ядро HTTP, это основной элемент, через который проходят все запросы. Здесь могут быть настроены обработка ошибок, ведение журнала и другие задачи, выполняемые до обработки запроса.

После загрузки ядра, на основе конфигурационных файлов определяется общий набор посредников, через который пройдут все приходящие запросы. Например, при включенной отладке, приложения использует посредника вывода и обработки ошибок. Он отобразит путь и тип ошибки для последующего её исправления.

Далее ядро загружает поставщиков услуг (service provider). Их список располагается в файле *config/app.php* и представляет набор библиотек и расширений, необходимых для работы приложения.

Когда все поставщики услуг будут загружены, запрос передается маршрутизатору. Он отправляет запрос на маршрут или контроллер, а также запускает посредников (middleware), прописанных для текущего маршрута.

HTTP-запрос попадает в контроллер или сразу возвращает представление, опуская контроллер. Контроллер выполняет определённые действия и отправляет данные в представление. Представление вставляет данные в свой шаблон, который будет отображен пользователю, в качестве ответа на запрос.

2.2 Библиотеки и плагины

Для реализации отдельных элементов системы, были использованы сторонние библиотеки и плагины, они включают набор готовых решений конкретных задач и позволяют уменьшить время разработки приложения.

При создании приложения возникли такие задачи, как обеспечение возможности редактирования статичных страниц с информацией администратором, создание QR-кодов и запись кодов от мероприятий, запись и формирование данных в файлы различных форматов документов и вывод данных о местоположении с помощью карт. Все описанные проблемы решены с использованием сторонних библиотек, некоторые представлены далее.

2.2.1 Библиотека для создания QR-кодов (SimpleQRCode)

Одна из возможностей приложения заключается в генерации QR-кодов, которые хранят коды от мероприятий. Для реализации была выбрана библиотека SimpleQRCode. Она является адаптированной под Laravel версией библиотеки BaconQRCode. Для использования библиотеки, регистрируется поставщик услуг, который позволит использовать класс SimpleQRCode в последующем коде приложения.

Создание QR-кода происходит через обращение к методу generate (“”) класса SimpleQRCode, где в скобках метода указывается строка, которая должна быть закодирована. Библиотека включает большое количество настроек для QR-кодов, подходящих под различные требования. При генерации она поддерживает следующие параметры:

- Size – настаивает размер изображения QR-кода, указывается в пикселях.
- BackgroundColor – настраивает цвет фона QR-кода.

- Merge – настраивает изображение в центре кода. В QR-код можно вставить изображение, например, логотип сайта.
- Format – настраивает формат результирующего изображения. Поддерживаются типы: png, eps, svg.

В результате использования, получается изображение в указанном формате, которое может быть выведено пользователю на странице или сохранено в файле.

2.2.2 Библиотека для редактирования информационных страниц сайта (TinyMCE Editor)

TinyMCE – это HTML-редактор, написанный на Javascript. Он предоставляет возможность редактирования HTML - разметки и включает широкий набор дополнительных функций по вставке и изменению содержимого страницы. TinyMCE обладает простым интерфейсом, который понятен для человека. С его помощью обычные пользователи могут создавать и изменять статьи и информационные страницы сайта.

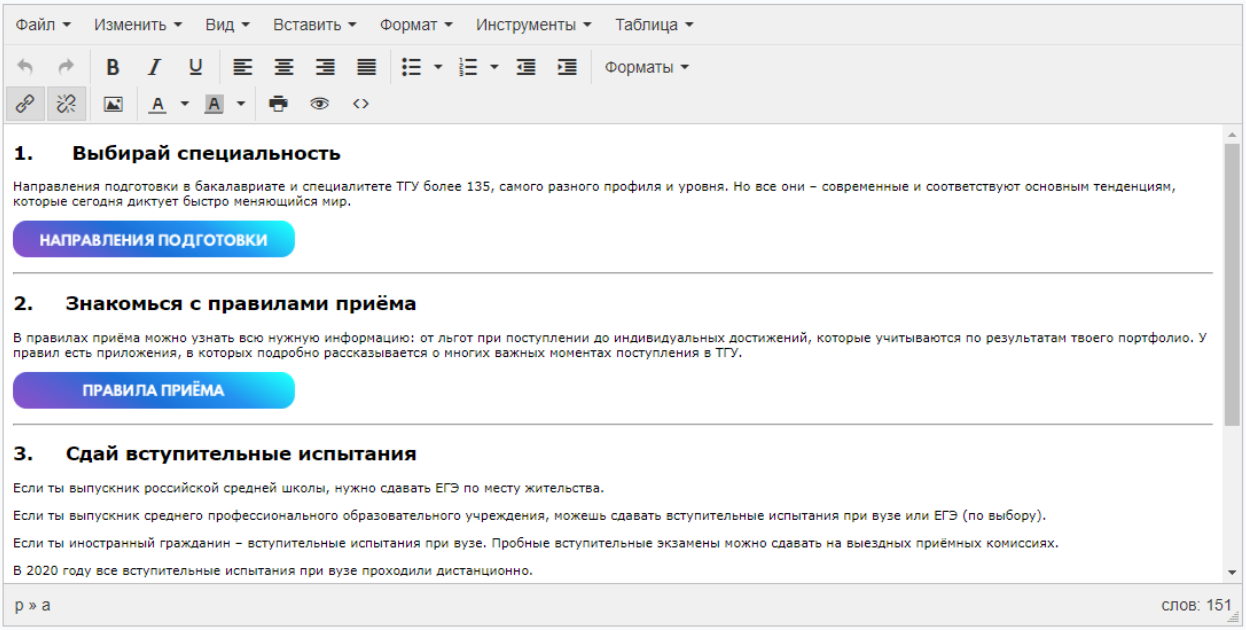


Рисунок 13 - Текстовый редактор TinyMCE

Функционал плагина включает:

- Изменение текста - шрифт, выравнивание, отступ, цвет, стили. Позволяет настроить стиль и формат для символов, аналогичный

текстовому редактору. В результате, текст его разметка, переводится в html, для вывода в качестве страницы или блока.

- Вставка изображений. В плагине доступна вставка изображения в текст, с указанием размеров. Для использования данной функции необходимо дополнительно подключить файловый менеджер, в качестве места хранения добавленных изображений.
- Вставка таблиц.
- Вставка ссылок.
- Правка HTML - разметки. Есть несколько вариантов редактирования: в формате текстового редактора или html – кода. В любой момент пользователь может сменить формат на html – код, для написания стилей для элементов текста на языке CSS.

Данный плагин упрощает работу администраторов сайта в изменении статичных страниц и блоков.

3. Реализация

На этапе реализации приложения были удовлетворены все поставленные требования и создан сайт «Личный кабинет школьника» для ТГУ. Сайт включает раздел мероприятий, профиль пользователя, магазин предметов, а также административную панель для управления сущностями сайта. Рассмотрим то, как разработаны основные части приложения.

Исходя из особенностей фреймворка Laravel в диаграмме классов произошли следующие изменения, представленные на изображении ниже.

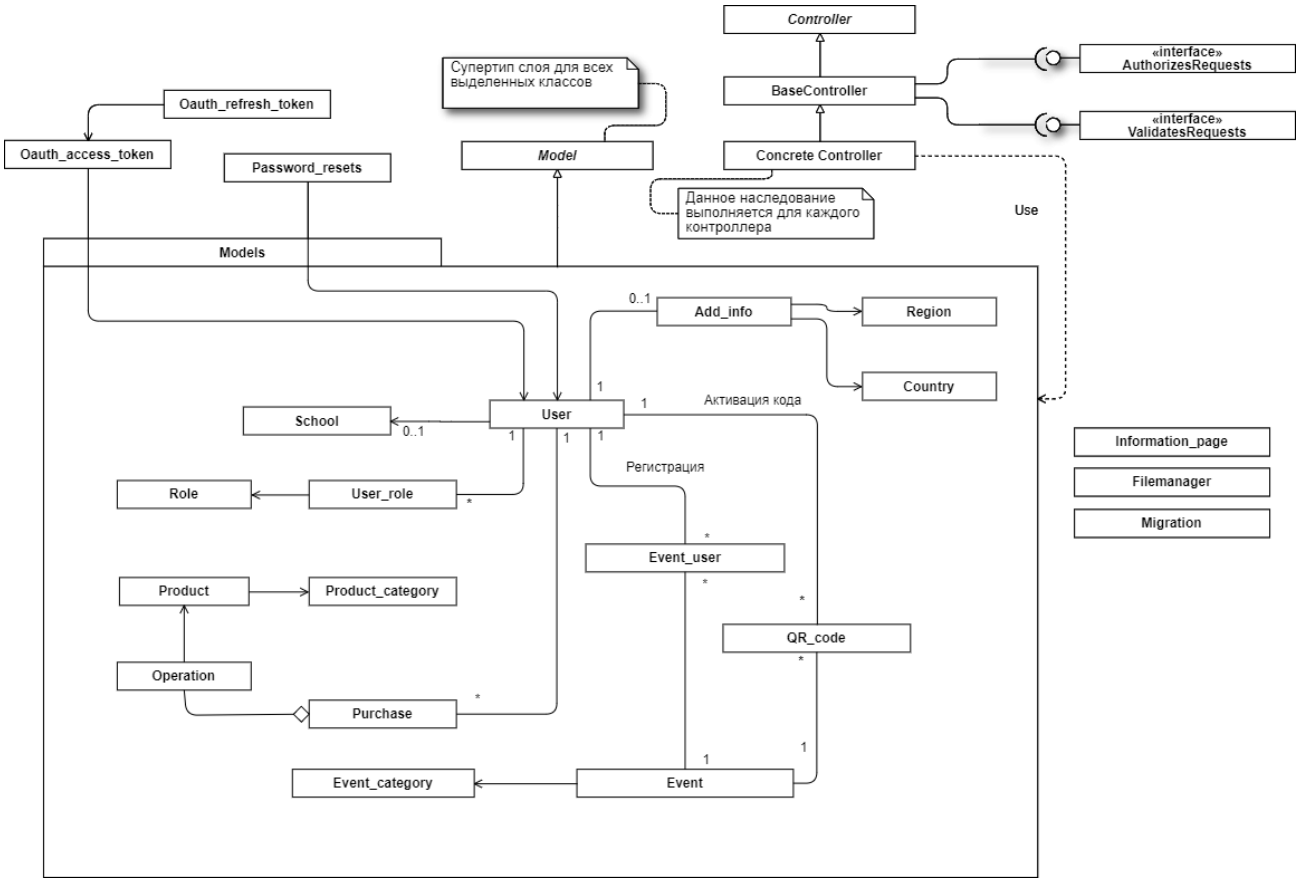


Рисунок 14 – Итоговая диаграмма классов веб-приложения

Все основные модели в системе наследуются от класса Model, это абстрактный класс, который содержит набор функций для взаимодействия с данными, в том числе и подключает возможности Eloquent ORM. Данный класс является суперклассом для всех моделей приложения.

На диаграмме отображена связь контроллеров приложения («Controller»). Все контроллеры наследуются от класса BaseController, через него происходит подключение интерфейсов системы, таких как AuthorizesRequests или ValidatesRequests. Представленные интерфейсы

обеспечивают проверку авторизации пользователя, отправляющего запрос и проверку данных входящего запроса.

В свою очередь, BaseController наследуется от абстрактного класса Controller, он предоставляет возможность использовать посредников и содержит обработку некоторых ошибок.

Контроллеры при обработке запросов могут вызывать различные классы для получения данных из БД.

Связи многие-ко-многим приведены к один-ко-многим с помощью дополнительных классов, например, связь Мероприятие-пользователь реализованное классом Event_user.

Создан класс для хранения данных об информационных страницах, они содержат HTML-разметку и название.

Подключен файловый менеджер, который управляет файлами в приложении и отвечает за их хранение и преобразование. Его класс обладает данными о формате файлы, ссылке на него, а также размерами и весом.

Так как в будущем планируется создание мобильного приложения, добавлен доступ к авторизации с помощью токенов (Oauth_access_token). При авторизации через сторонние приложения, система выдает токен и использует его для определения пользователя.

3.1 Создать мероприятие

Мероприятие – основная сущность приложения и процесс его создания является основополагающим. Создание мероприятия открывает возможность просмотра информации и регистрации на него. Оно доступно только администраторам. Ниже описана работа приложения при выполнении варианта использования «Создать мероприятие».

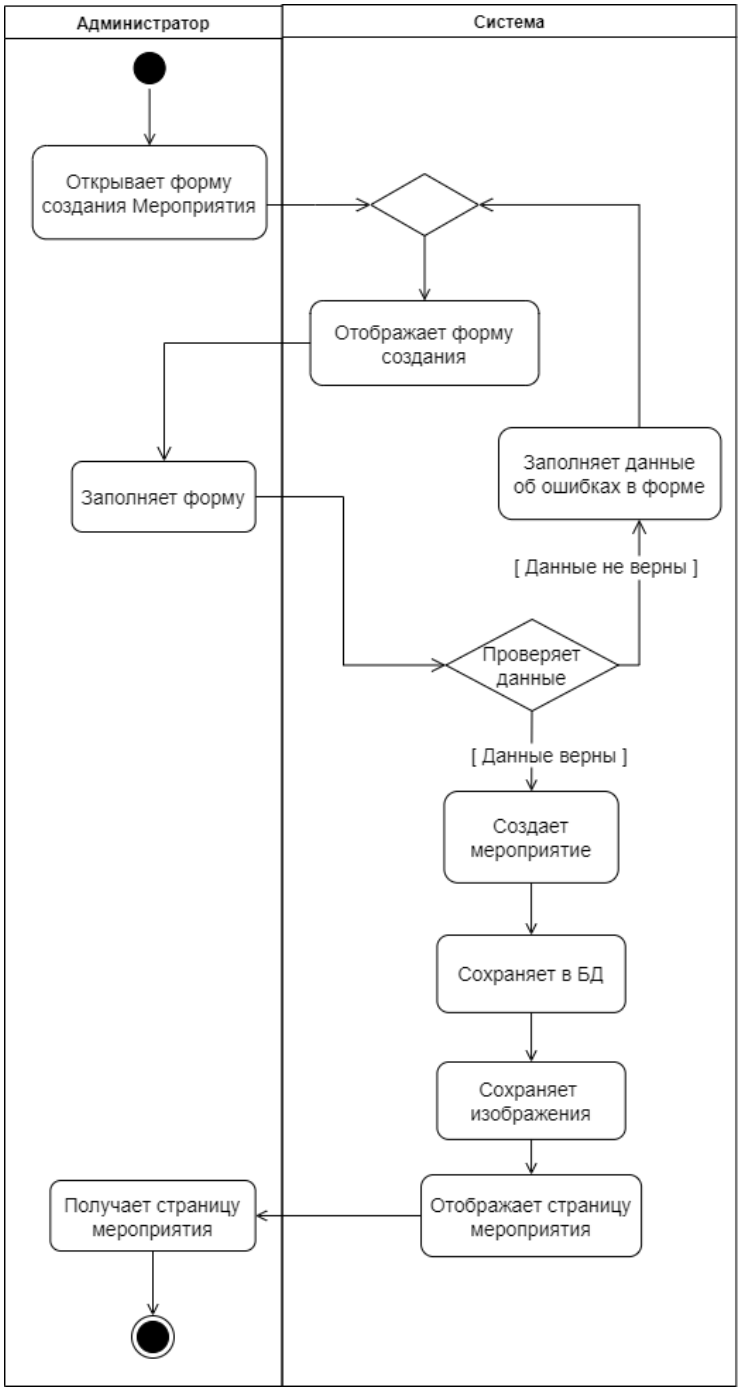


Рисунок 15 - Диаграмма активности варианта использования "Создать мероприятие"

Изображение выше описывает работу системы и её взаимодействие. Пользователь заполняет форму, которую система проверяет и, в результате, добавляет новое мероприятие в БД. Рассмотрим то, как выглядит этот вариант использования для администратора.

Первым шагом администратор, находящийся странице мероприятий, открывает форму создания мероприятия.

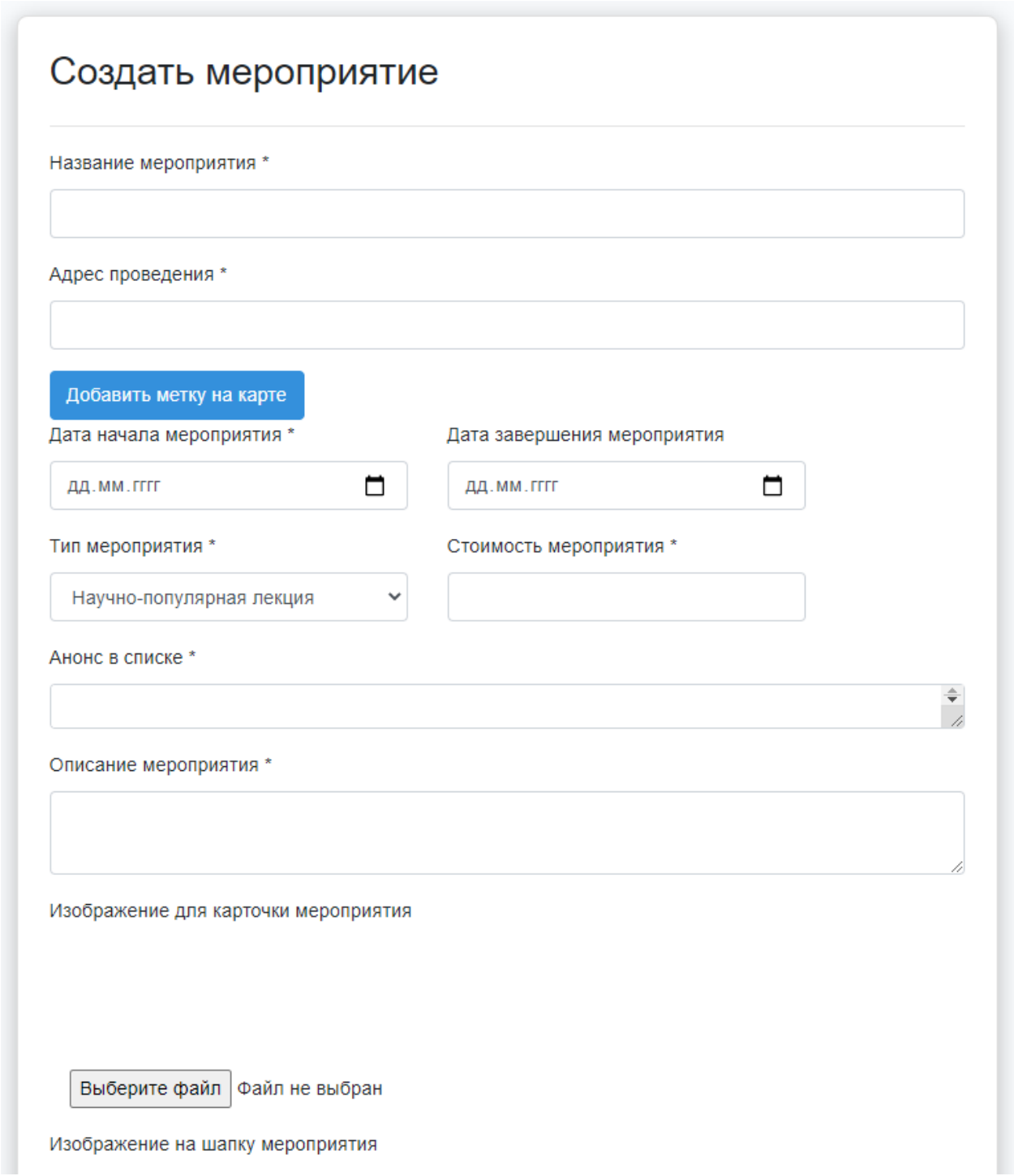


Рисунок 16 - Форма создания мероприятия

Он заполняет форму для создания мероприятия и отправляет её. Система проводит проверку данных. В случае неверных значений, форма будет возвращена системой, с указанием неверно заполненных полей.

Далее система записывает данные из формы в новое мероприятие и сохраняет его. В завершение пользователь перенаправляется на страницу созданного мероприятия.

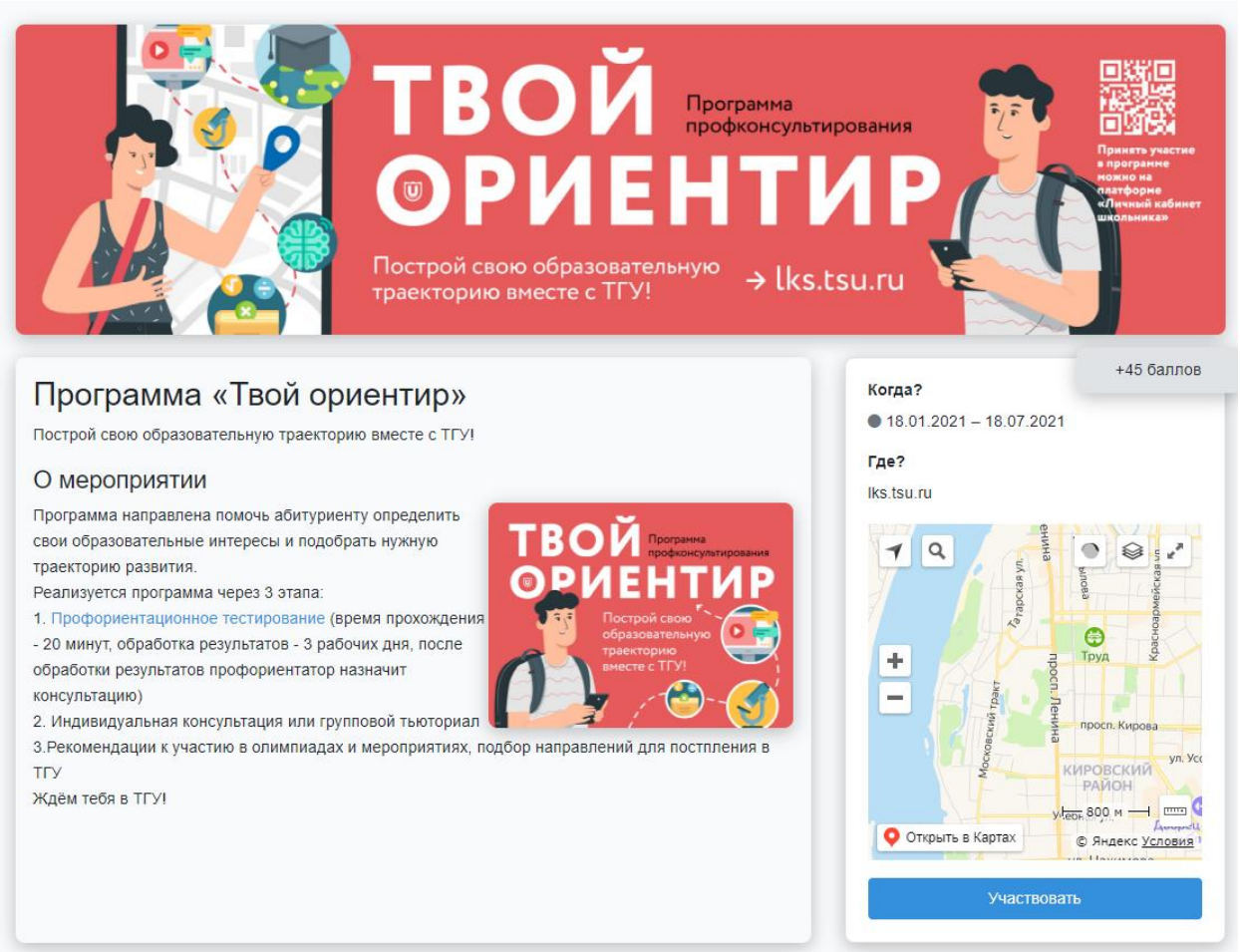


Рисунок 17 – Страница мероприятия

3.2 Сгенерировать коды для участников

Вариант использования “Сгенерировать коды для участников”. Выполнив его, администратор получает файл с кодами для подтверждения посещения мероприятия. Эти коды в последующем передаются пользователю для их использования. Рассмотрим его реализацию на диаграмме последовательности представленной ниже.

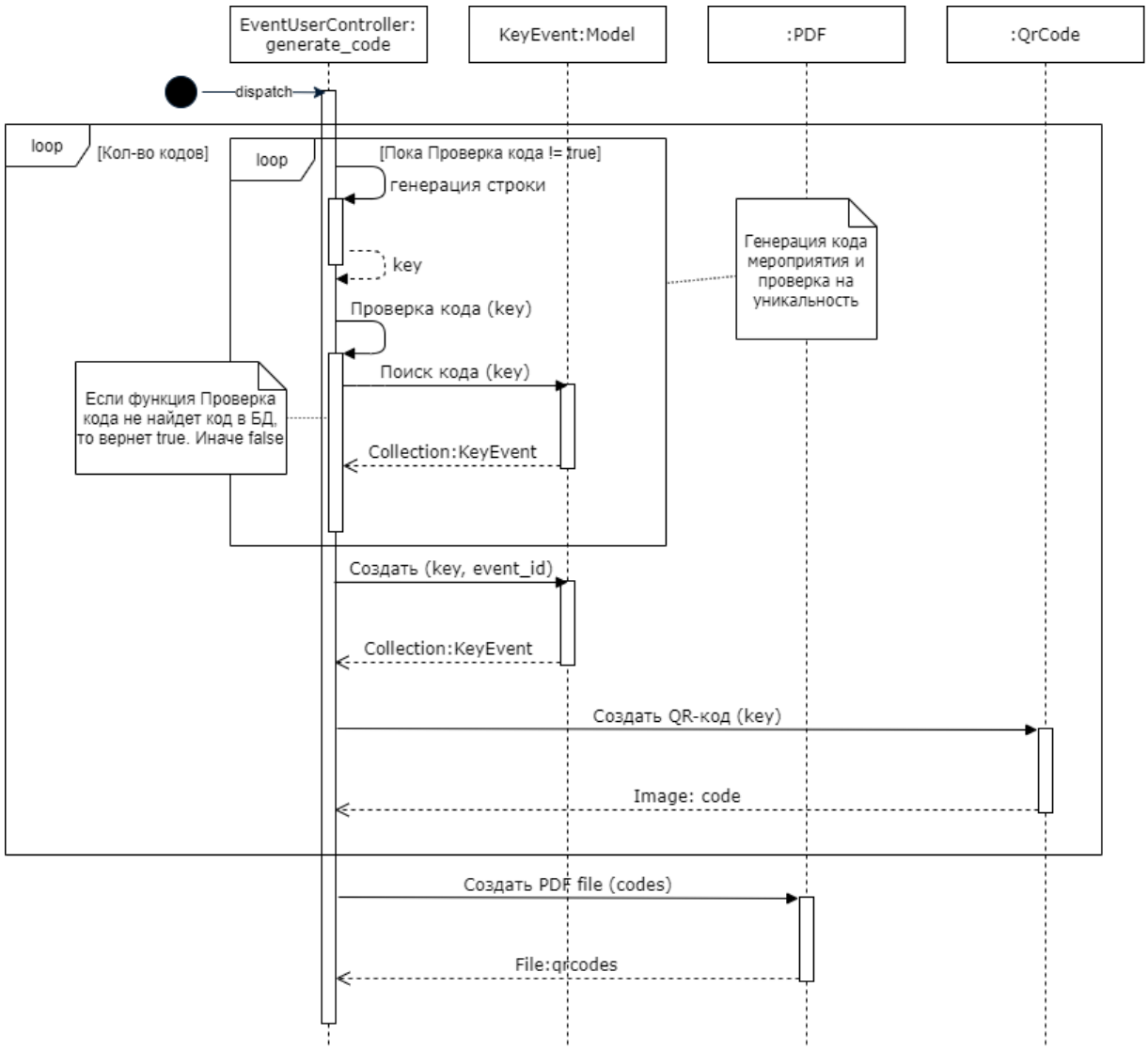


Рисунок 18 - Диаграмма последовательности Описание генерации кодов для участников

Функция получает на вход id мероприятия и необходимое количество кодов. В цикле генерируются коды мероприятий, их количество соответствует полученному из формы. Каждый код проходит проверку на уникальность, если такой код еще не записан в системе, то создается класс KeyEvent в который записывается этот код и id мероприятия. После того как код прошел проверку, он записывается в QR-код. QR-коды создаются с помощью сторонней библиотеки SimpleQRCode. Все полученные QR-коды сохраняются в один файл и отправляются в качестве ответа администратору.

Так выглядит реализация со стороны системы. Администратор вариант использования «Сгенерировать коды для участников» видит следующим образом.

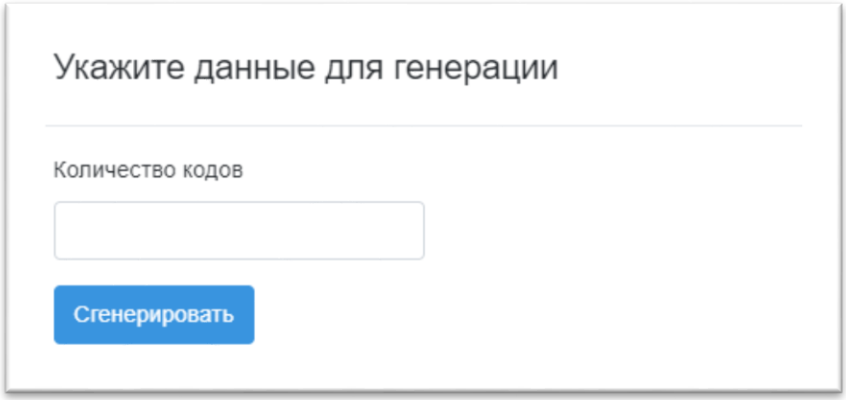


Рисунок 19 – Форма генерации кодов

Пользователь, с ролью администратор, на странице мероприятия открывает форму генерации QR-кодов. Форма представлена на рисунке выше, пользователь указывает количество и отправляет её. В результате, он получает файл с новыми QR- кодами для участников. Пример одного кода, представлен на скриншоте ниже.

Рисунок 20 – QR-код из файла кодов мероприятия

3.3 Подтвердить посещение мероприятия

Вариант использования “Подтвердить посещение мероприятия”. Данный вариант использования позволяет пользователю подтвердить посещение мероприятия и получить баллы. Подтверждение происходит на странице профиля пользователя.

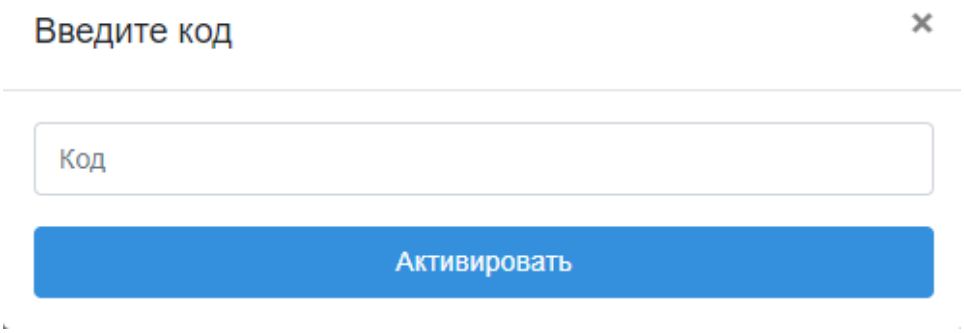


Рисунок 21 - Форма ввода кода мероприятия

Для выполнения этого варианта использования, он открывает форму активации кода в профиле, которая представлена выше. Вводит тот код, что записан в QR-коде, который выдал организатор мероприятия. В QR-коде записаны код подтверждения и ссылка на подтверждение. Пользователь может перейти по ссылке, и система автоматически проведет подтверждение и выдаст ответ. Он может ввести его самостоятельно, если это предпочтительнее. В результате, система возвращает сообщение об успешном подтверждении или об ошибке.

Рассмотрим то, как происходит обработка запроса со стороны системы. На диаграмме последовательности ниже, представлена реализация.

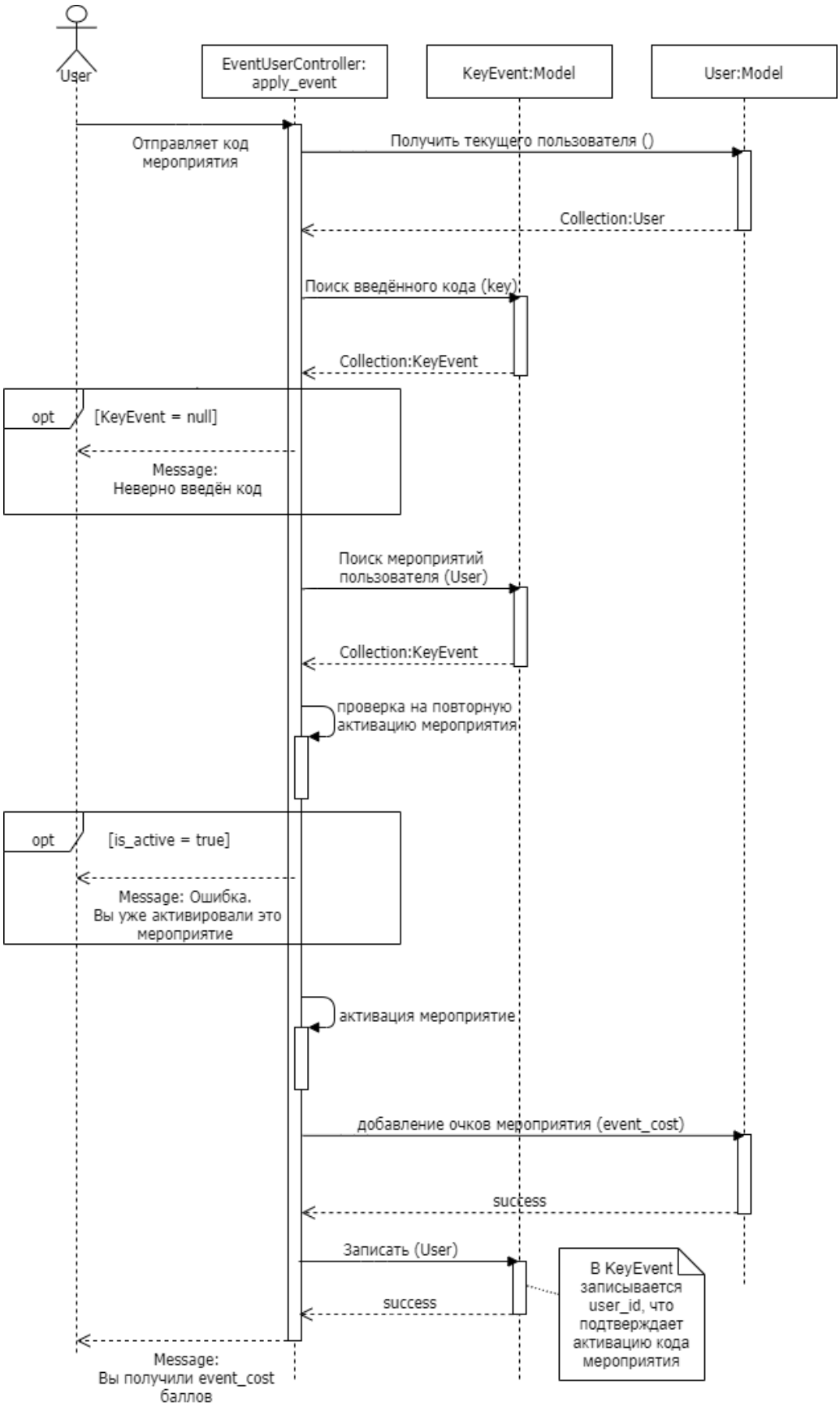


Рисунок 22 - Диаграмма последовательности Подтверждение посещения мероприятия

На вход поступает код мероприятия. Система определяет пользователя, отправившего запрос. Далее она находит код в БД, если код не найден система возвращает сообщение об ошибке. Когда код найден в базе, определяется к какому мероприятию он относится и происходит проверка мероприятий, которые подтвердил пользователь. Если пользователь уже подтверждал мероприятие, к которому относится введенный код, система вернет сообщение, что мероприятие уже подтверждено. Если это мероприятие подтверждается впервые, то система начислит баллы за посещение пользователю и сохранит информацию о подтверждении в модели KeyEvent.

3.4 Создать предмет магазина

Магазин предметов является еще одной частью системы, которая обладает широким функционалом. Она отвечает за приобретение и получение предметов магазина. Предмет магазина, товар – это то, вокруг чего построена работа магазина. Рассмотрим вариант использования создать предмет магазина.

Описание варианта использования представлена на диаграмме взаимодействия ниже. Пользователь с ролью администратор переходит на страницу создания товара для магазина. Он заполняет информацию о товаре, его количество и изображения товара, далее отправляет форму. Система проверяет введенный данные и при ошибке возвращает пользователю на исправление. Если данные введены верно, система создает новый товар и сохраняет его. Также, если указано количество товара, создает запись типа Операция. Класс Операция предназначен для учета количества товара и его движений. Она бывает двух типов – Приход или Расход. После создания, система перенаправляет пользователя на страницу магазина и сообщает об успешном создании товара.

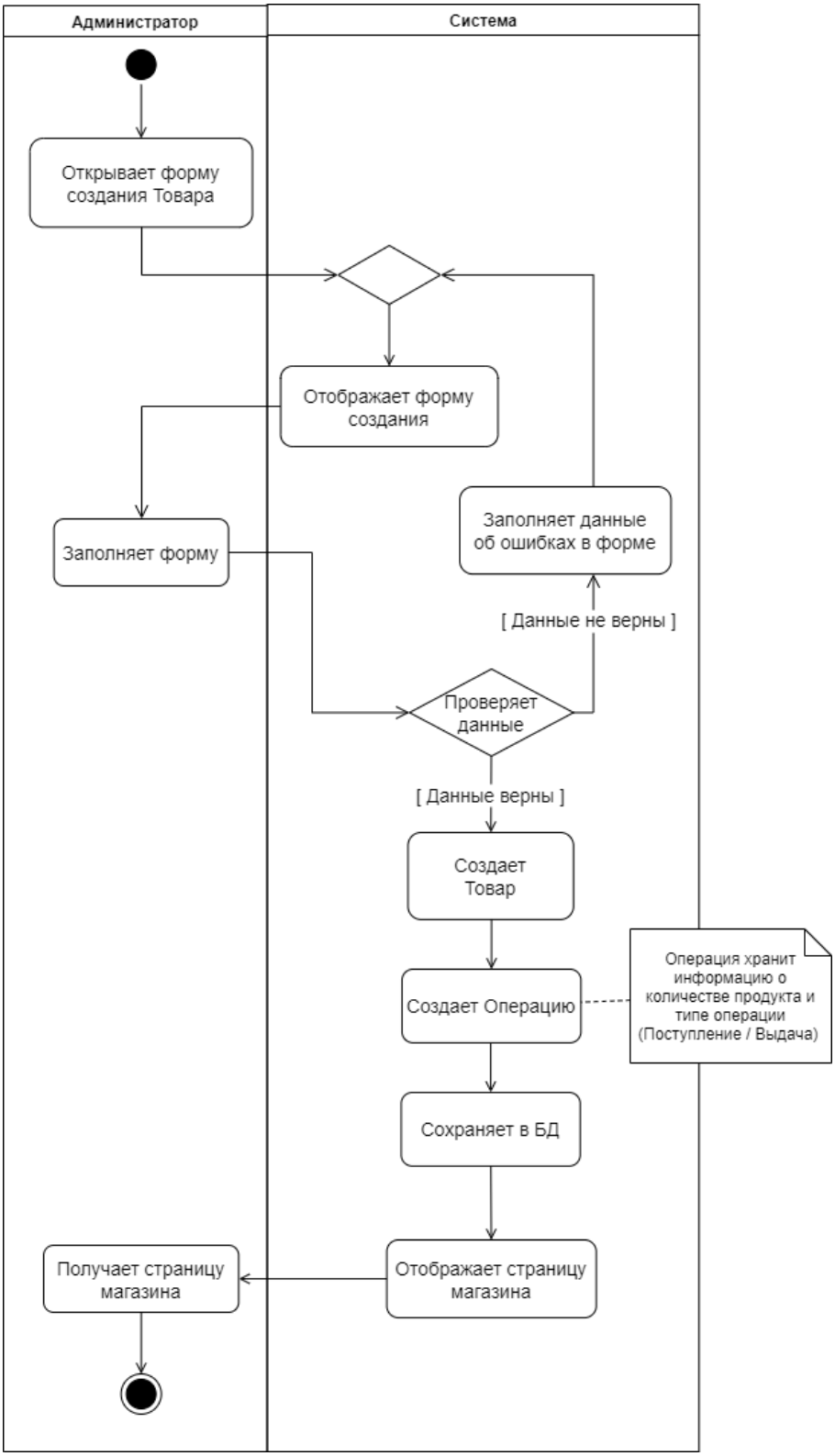


Рисунок 23 – Диаграмма активности варианта использования «Создать предмет магазина»

Рассмотрим вариант использования со стороны пользователя. Администратор открывает форму создания, представленную на изображении ниже.

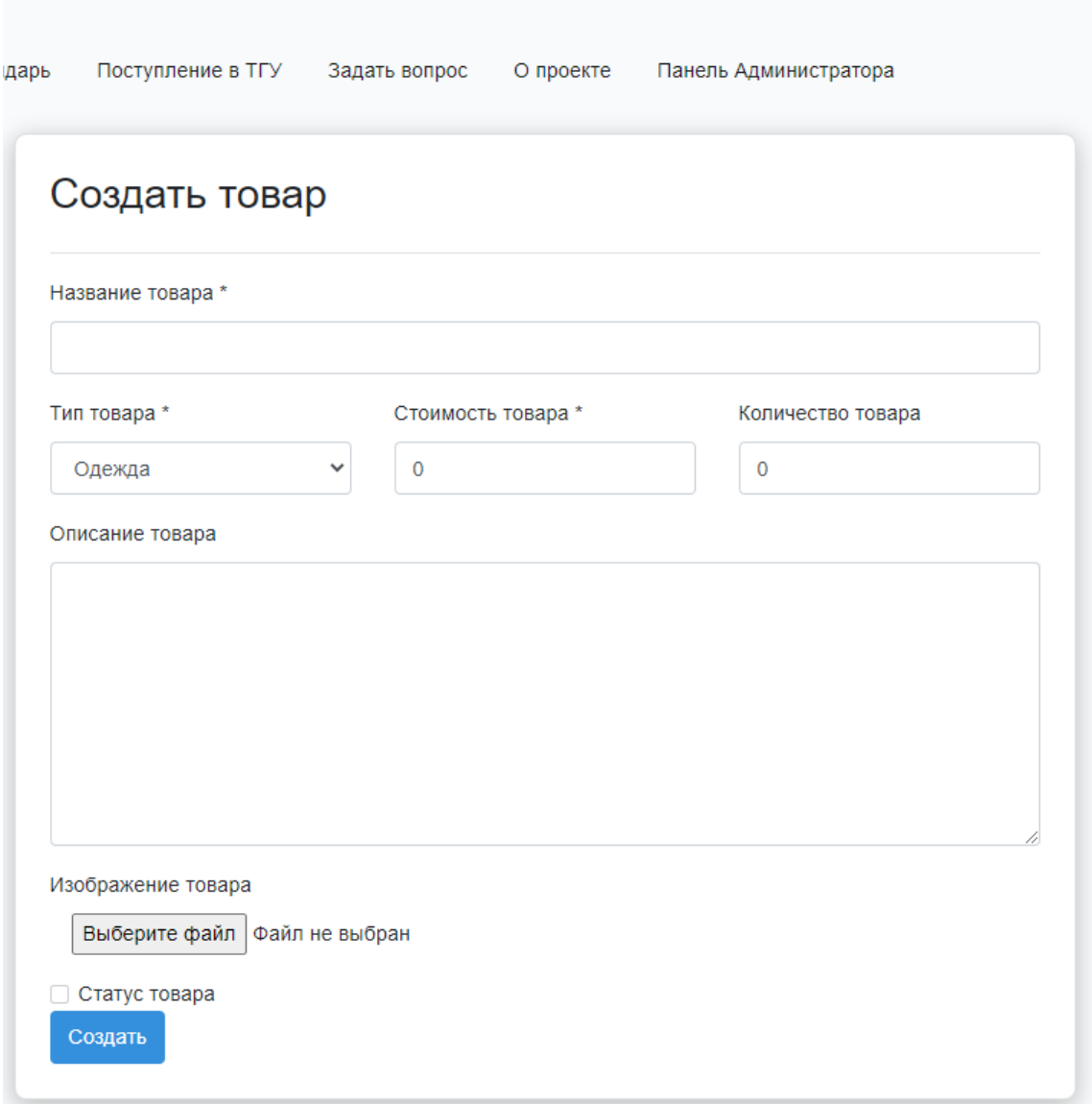


Рисунок 24 – Форма создания товара

Заполнив форму, он нажимает кнопку «Создать». Приложение обрабатывает полученные данные и создает новый товар. Созданный товар отобразится в магазине и списке товаров в панели администратора.

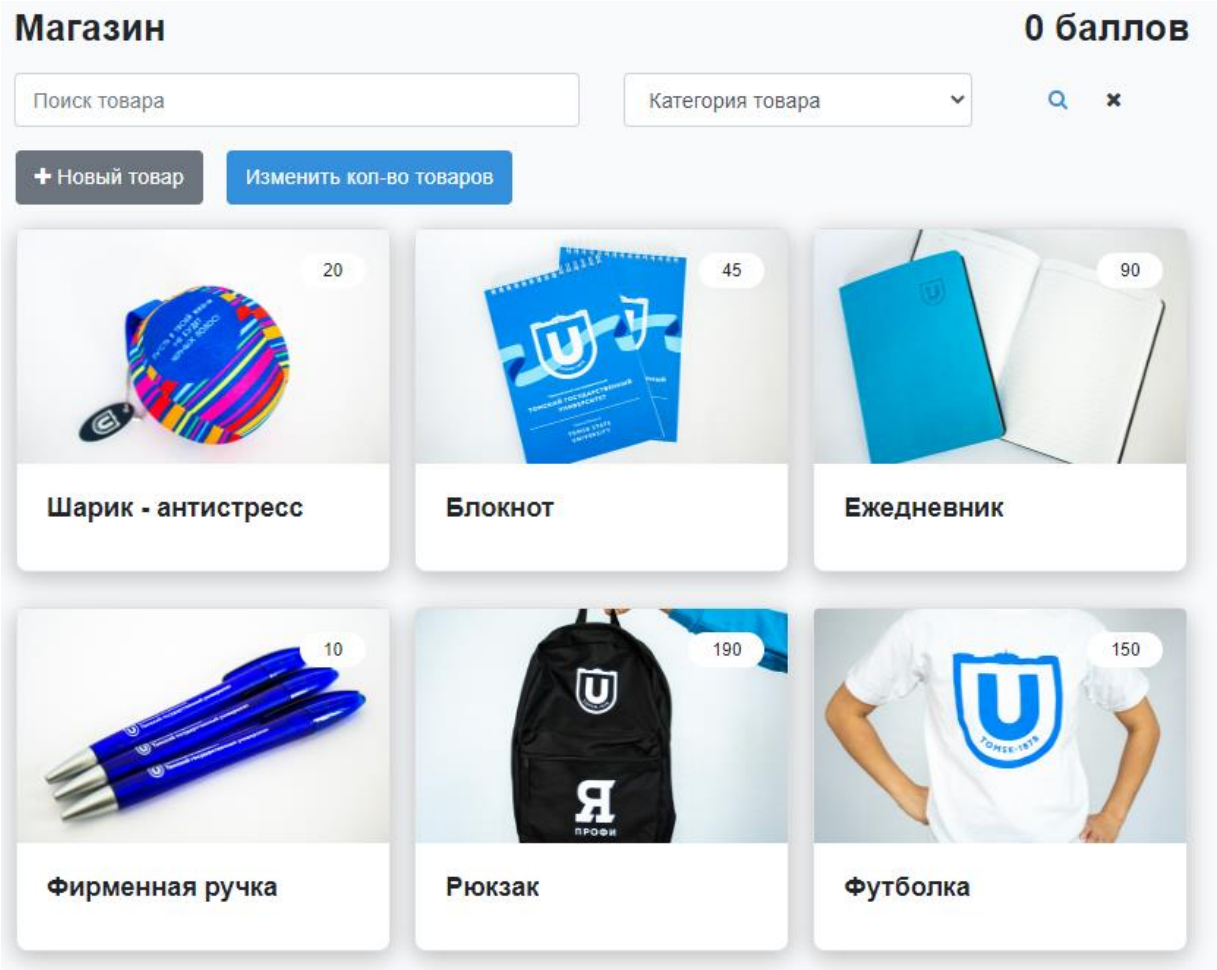


Рисунок 25 – Страница магазина

3.5 Приобрести предмет магазина

Создание товара подразумевает его продажу и выдачу пользователям. Рассмотрим приобретение товара пользователем. Авторизованный пользователь открывает магазин предметов и покупает желаемый товар. Система получает запрос и выполняет его в соответствии с описанием, представленным на диаграмме ниже.

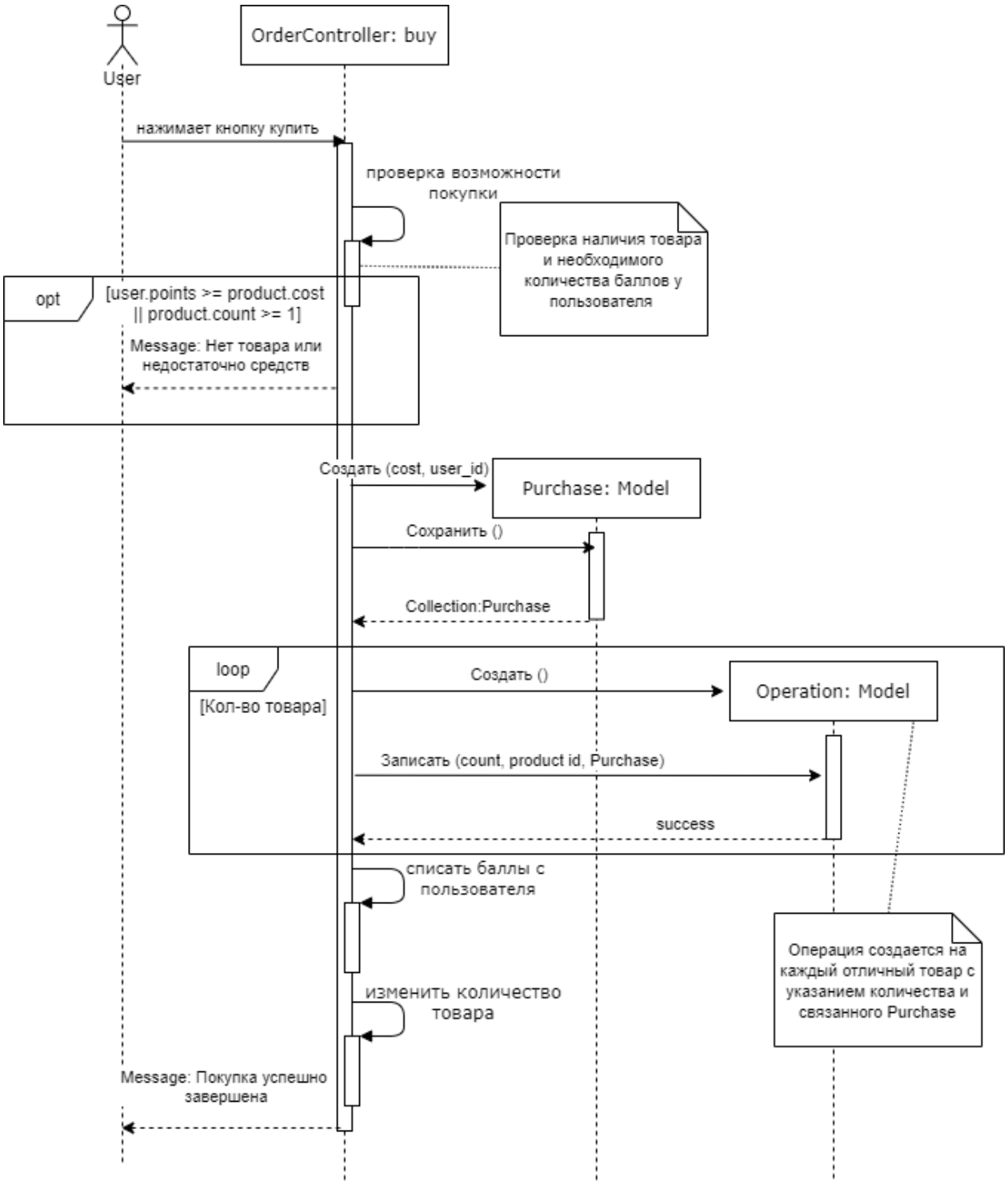


Рисунок 26 – Диаграмма последовательности «Приобрести предмет магазина»

Пользователь инициирует выполнение покупки нажав на соответствующую кнопку. Приложение получает на вход выбранный товар и пользователя, купившего его. Первоначально происходит проверка возможности покупки товара. Количество баллов пользователя должно быть больше или равным стоимости товара. В случае если баллов недостаточно или

товар отсутствует в продаже, система выведет сообщение об ошибке. Иначе система записывает информацию о товаре, количестве и покупателе в БД. После этого списывает баллы за покупку с пользователя и изменяет количество товара в магазине. В результате, пользователь получает сообщение об успешном совершении покупки. Купленный товар будет отображаться в разделе награды, в профиле.

ЗАКЛЮЧЕНИЕ

В рамках данной работы решены следующие поставленные задачи

- Собраны и формализованы требования к разработанной системе
- Спроектирована архитектура приложения «Личный кабинет школьника»
- Приложение «Личный кабинет школьника» разработано и находится по адресу: http://lks.tsu.ru/

СПИСОК ИСПОЛЬЗОВАНОЙ ЛИТЕРАТУРЫ

1. Архитектура корпоративных программных приложений. / Мартин Фаулер Издательский дом "Вильяме' 2006. 541с.
2. Руководство по PHP, 2001-2021. – URL:https://www.php.net/manual/ru/ (дата обращения: 12.11.2020)
3. Справочник Bootstrap 4 – URL: https://bootstrap-4.ru/ (дата обращения: 25.02.2021)
4. Язык UML. Руководство пользователя, 2-е издание / Издательство: ДМК Пресс. 2007. 496 с.
5. Laravel 7 documentation, 2011-2021 – URL: https://laravel.com/docs/7.x (дата обращения: 08.12.2020)
6. Laravel. Полное руководство [2-е издание] / Мэтт Стаффер Издательство: Питер 2020. 514 с.
7. The Clean Architecture in PHP / Кристофер Уилсон. – URL : https://leanpub.com/cleanphp (дата обращения: 28.01.2021)
8. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с.,

АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

Отчет о проверке на заимствования №1

Автор: Карагеоргий Алексей
Проверяющий: Карагеоргий Алексей (red_bull_135@mail.ru / ID: 8153991)

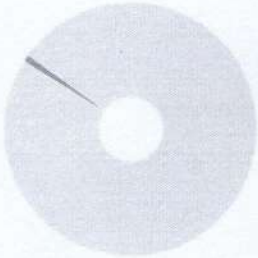
Отчет предоставлен сервисом «Антиплагиат» - users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 3
Начало загрузки: 08.06.2021 07:23:30
Длительность загрузки: 00:00:01
Имя исходного файла: ВКР ЛКШитог.pdf
Название документа: ВКР ЛКШитог
Размер текста: 52 кБ
Символов в тексте: 52893
Слов в тексте: 5933
Число предложений: 462

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 08.06.2021 07:23:32
Длительность проверки: 00:00:02
Комментарии: не указано
Модули поиска: Интернет



ЗАИМСТВОВАНИЯ	САМОЦИТИРОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
0,5%	0%	0%	99,5%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Актуален на	Модуль поиска	Комментарии
[01]	0,26%	http://vital.lib.tsu.ru/vital/access/services/Download/vital:12348/SOURCE01 http://vital.lib.tsu.ru	07 Сен 2020	Интернет	
[02]	0,25%	Гуманитарный факультет http://week-science.spbstu.ru	08 Янв 2021	Интернет	
[03]	0%	8_121_74_0_0.600_76999972 гудлиф - ремесло программиста. практика написания хорошего кода.pdf.zip/Гудлиф - Ремесло программиста. Практика написания хорошего кода.pdf http://inethub.olvi.net.ua	22 Янв 2021	Интернет	Источник исключен. Причина: Маленький процент пересечения.