**Глава 2. Проектирование и реализация программного продукта**

1. **Выбор методов решения задания**

Создание программы в направлении веб-API, разделение на back end и front end, что помогает уменьшить взаимозависимость двух частей:

- База данных: Использование реляционной базы данных SQL (MySQL);

- Back end: использована Spring Boot для написания API, использован JPA для сопоставления класса (объектно-ориентированный подход) в коде с таблицей (реляционный подход) SQL;

- Front end: интерфейс cоздано с помощью HTML, CSS. Взаимодействие с Backend API с помощью библиотеки jQuery (в частности, AJAX).

Model – View – Controller (MVC) - это шаблон проектирования, целью которого является разделение на интерфейс и код для упрощения управления, разработки и обслуживания. MVC делит программное приложение на 3 взаимодействующие части:

- Модель - это класс, содержащий информацию об объекте (данные), который взаимодействует с Базой Данных. Отвечает за моделирование объектов;

- Представление - это интерфейс системы, непосредственно взаимодействующий с пользователем;

- Контроллер получает запрос от пользователя. Он использует модель и представление для обработки и возврата результатов пользователю.

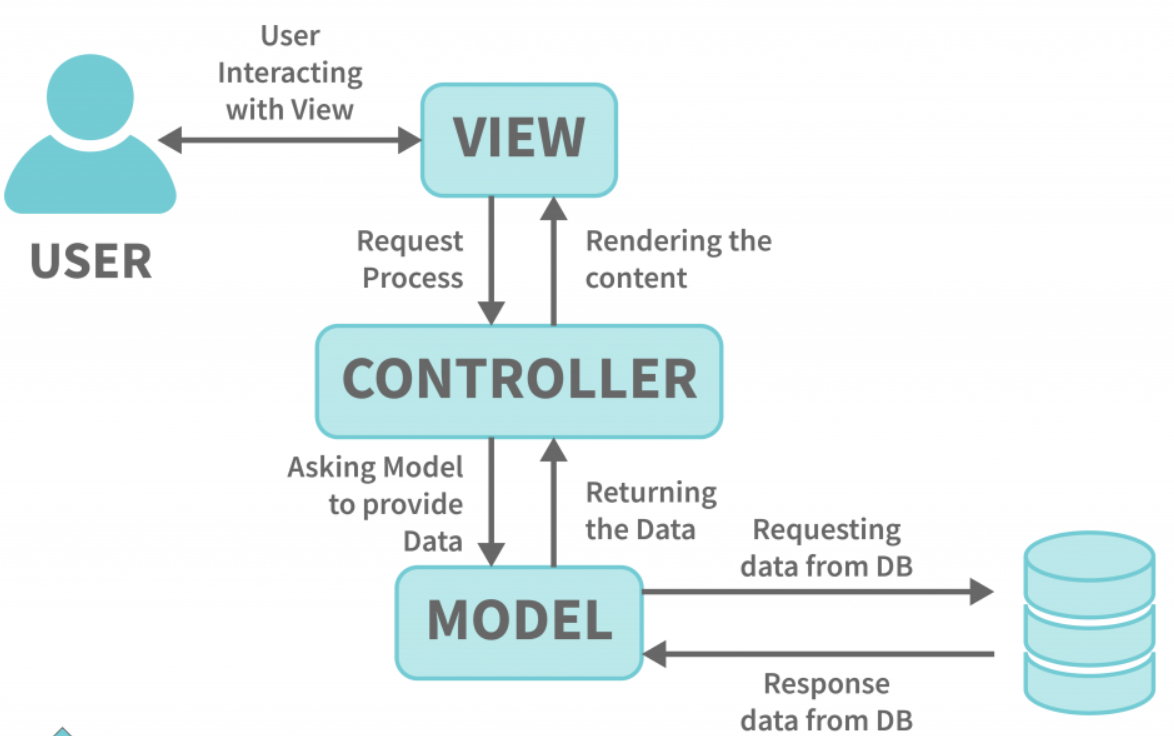


Рисунок - Работа MVC

При этом, представление заключается в front end, а модель и контроллер заключаюся в back end.

Spring Boot – инструмент фреймворка Spring для написания приложений на Spring с минимальной конфигурацией. Данный инструмент также имеет встроенный контейнер сервлетов (Tomcat по умолчанию) что значительно упрощает запуск приложения. Веб приложения на Spring Boot запакованы в jar файл, что позволяет запускать их как обычные java приложения.

1. **Разработка диаграммы классов программы**

Одна из причин использования объектного подхода — близость к реальному миру. Такие сущности, как предметы, вопросы и экзамены, очень знакомы и близки, можно легко абстрагировать их в классы, чтобы из каждого класса можно создавать объекты с похожими свойствами и методами, соответствующие этой сущности в реальной жизни.

По вариантам использования системы можно определить требуемые объекты системы, потом определить отношения между объектами. Основываясь на этом, существуют 8 классов (и каждому классу соответствует таблица в базе данных). В соответствии с перечисленными требованиями на рисунке х представлена концептуальная диаграмма классов. В этом использована агрегация — отношение когда один объект является частью другого. Заметно, что в случае агрегации целое хоть и содержит свою составную часть, время их жизни не связано.

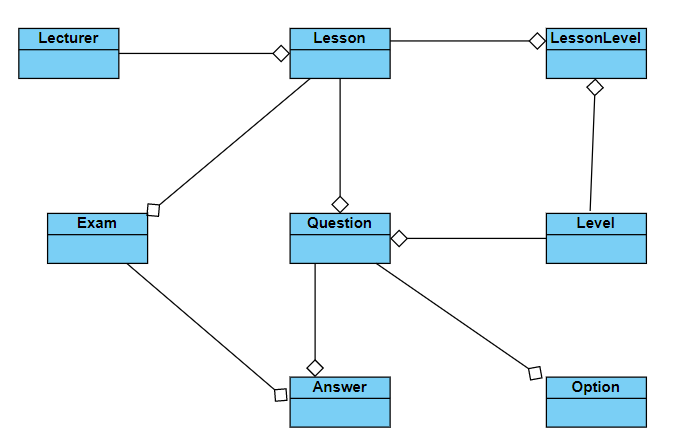


Рисунок - Концептуальная диаграмма классов

На следующем шаге определёны свойства и методов каждого объекта. Из вышеприведенного анализа отношения между объектами системы показаны на рисунке х.

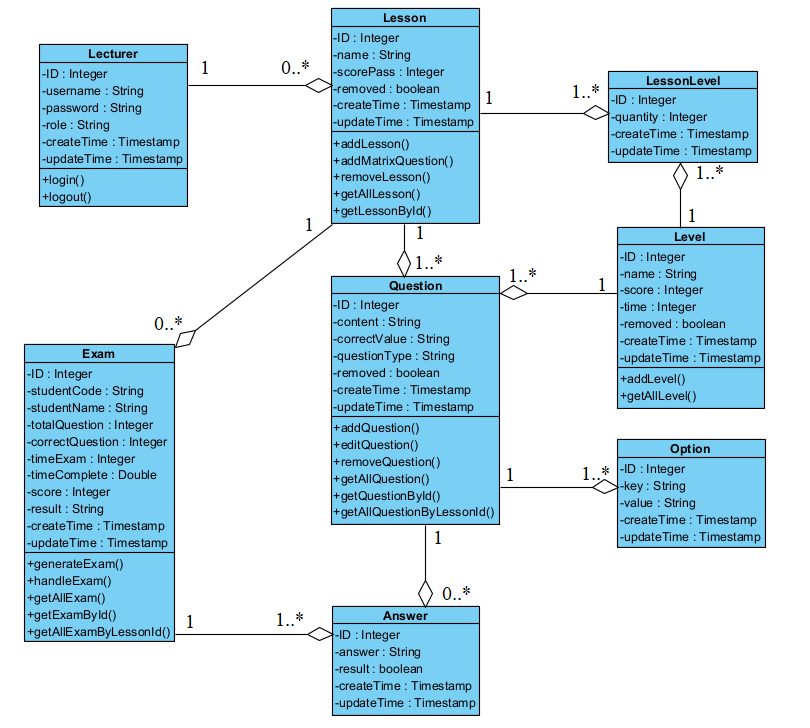


Рисунок – Диаграмма классов

1. **Процесс обработки информации и анализ структуры данных**

С помощью процесса анализа диаграммы классов можно начинать построить базы данных. Техническое задание проанализовано для ясности, какие данные должны находиться в базе данных.

Необходимо связывать таблицы вместе, чтобы корректно импортировать и получать данные. Использована связь «Один ко многим» (с обязательной связью и с необязательной связью) в таблицах.

- Таблица «lecturer»: Надо связывание между таблицами «lecturer» и «lesson». Действительно, преподаватель может научить много предметов. Поэтому, связь между таблицами «lecturer» и «lesson»: 1 - N;

- Таблица «lesson»: В каждом предмете есть банк вопросов, уровни самого этого предмета и экзамены. Итак, связи между таблицами «lesson» - «question», «lesson» - «lessonlevel», «lesson» - «exam»: 1 – N;

- Таблица «lessonlevel»: связь между таблицами «lessonlevel» - «level»: 1 – N;

- Таблица «level»: Некоторые вопросы относятся к определённому уровню, поэтому связь между таблицами «level» - «question»: 1 – N;

- Таблица «question»: поскольку каждый вопрос задан с множественными выборами, связи между таблицами «question» - «option\_», «question» - «answer»: 1 – N;

- Таблица «answer»: Некоторые ответы относятся к определённому экзамену, поэтому связь между таблицами «answer» - «exam»: 1 – N.

Схема базы данных показана на рисунке х.

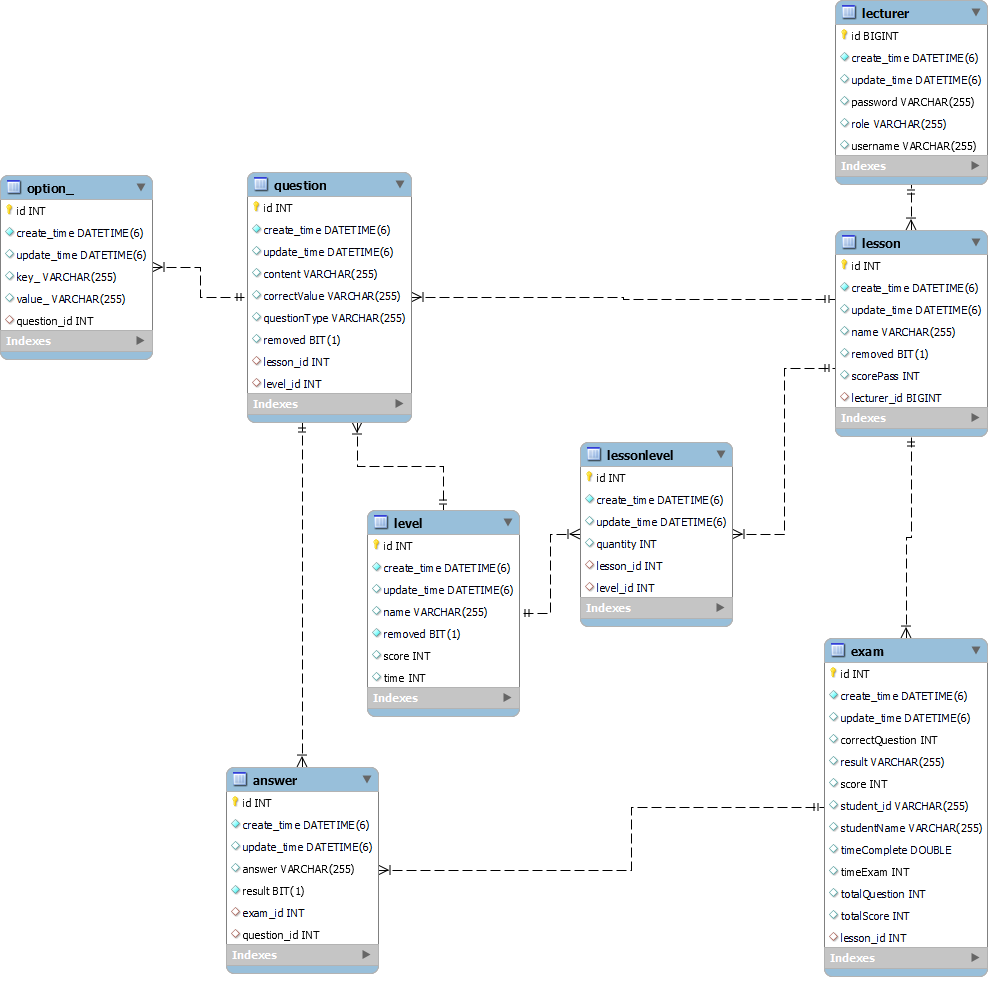


Рисунок - Схема базы данных

После процесса связываний между таблицами, начинать построить часть Back End по следующим шагам:

- Создать пакет сущностей (package entity), затем создать вышеуказанные классы объектов;

- Создать интерфейсы репозитория (interface repository) для взаимодействия с базой данных (database);

- Cоздать контроллер, содержащий API (Application Programming Interface);

- В каждом API будет вызываться соответствующий сервис (service) для выполнения бизнес-логики (business logic) этого API.

1. **Разработка диаграммы вариантов использования**

Разработку спецификаций программного обеспечения начинают с анализа требований к функциональности, указанных в техническом задании. В процессе анализа выявляют внешних пользователей разрабатываемого программного обеспечения и перечень отдельных аспектов его поведения в процессе взаимодействия с конкретными пользователями. Аспекты поведения программного обеспечения были названы «вариантами использования» или «прецедентами» (use cases).

Согласно техническому заданию, это веб-приложение используется многими пользователями (преподаватель и иностранные студенты). Выполняемые функции каждого пользователя представлены в таблице х.

Таблица - Выполняемые функции каждого пользователя

|  |  |
| --- | --- |
| Для преподавателя | Для иностранных студентов |
| * Авторизация; * Создание, редактирование и удаление вопросов; * Установка количества вопросов по каждому тестированию; * Просмотр всех вопросов в базе знаний; * Просмотр результатов тестирования пользователей системы. | * Ввод информации (Ф.И.О, номер группы, номер зачётки); * Прохождение теста за определенной время; * Просмотр финального результата тестирования. |

Из этой таблицы, получена диаграмма вариантов использования. Диаграмма вариантов использования представлена на рисунке х.



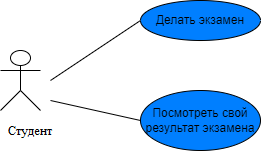


Рисунок - Диаграмма вариантов использования

На основе анализа диаграммы вариантов использования можно начинать следующий шаг проектирования: определить требуемые объекты системы.

1. **Разработка интерфейса пользователей**

Созданы интерфейсы с помощью HTML, CSS, JS. Используя такие библиотеки, как Bootstrap, jQuery помогает создавать более быстрые, полные и удобные интерфейсы для пользователей.

Front End общается и обменивается информациями с Back End через API. Обмен данными между двумя сторонами в формате JSON. Это распространенный формат данных в современных системах API. Когда пользователь взаимодействует с интерфейсом, Front End вызывает или отправляет необходимые данные в Back End через API, определенные в BE. Благодаря JSON, Front End и Back End могут понимать данные, которыми обмениваются друг с другом. Вызов API с использованием AJAX — это способ загрузки новых данных без перезагрузки всей страницы. Он взаимодействует только с элементами на странице и динамически обновляет данные для каждого элемента, с которым он взаимодействует.

После процесса анализа интерфейсных компонент, диаграммы состояний формы интерфейсов предстанвлены на рисунках х-у.

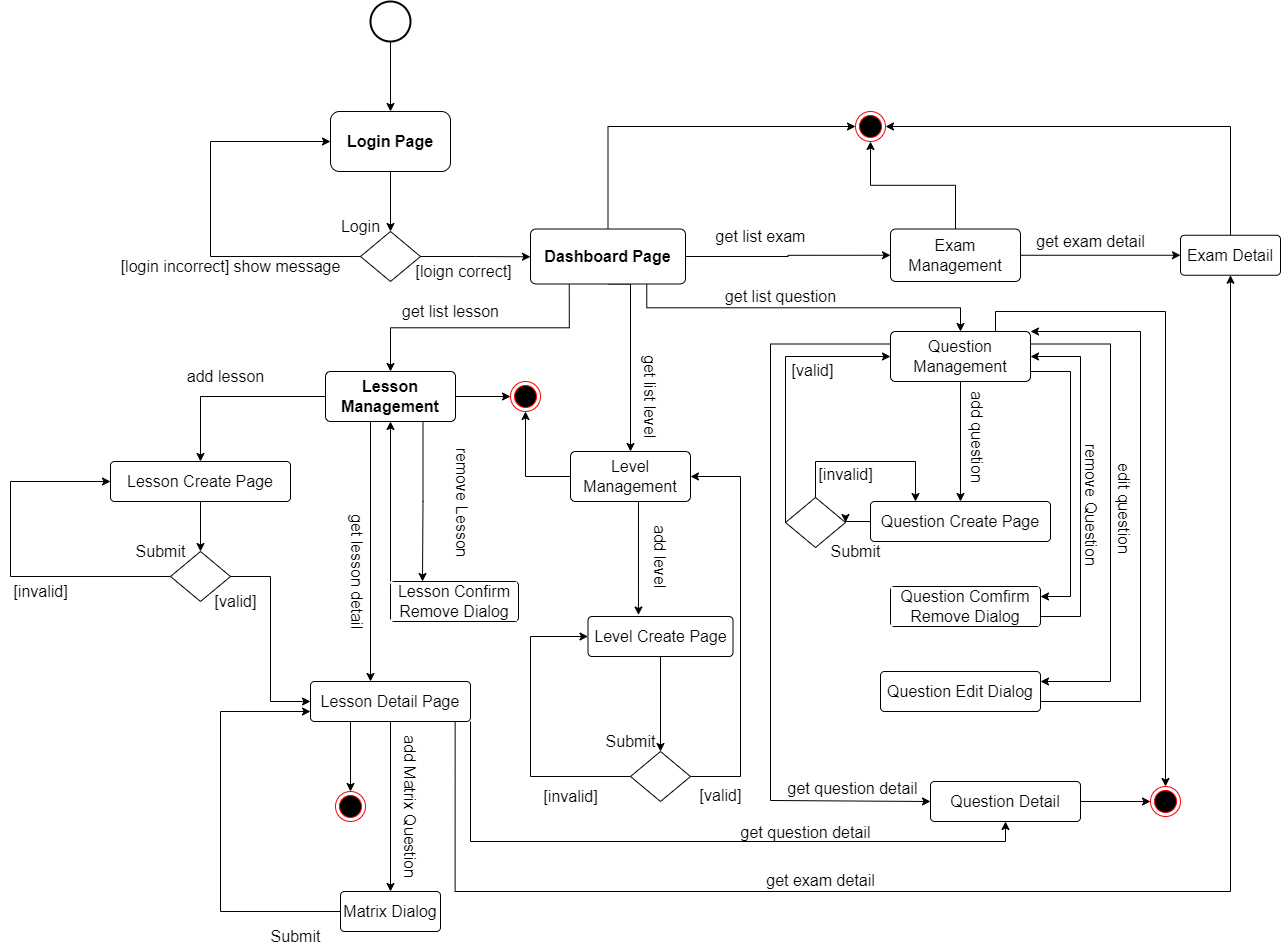


Рисунок - Диаграмма состояний формы интерфейсов преподавателя

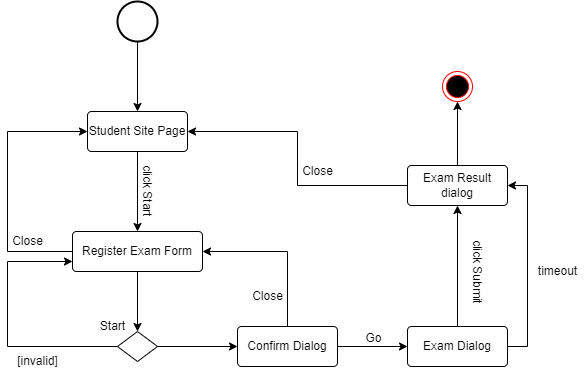


Рисунок - Диаграмма состояний формы интерфейсов студента

1. **Описание форм интерфейса**

Для преподаветеля, все возможности работы которого на некоторых страницах и их формы интерфейсов представлены на рисунках х-у.

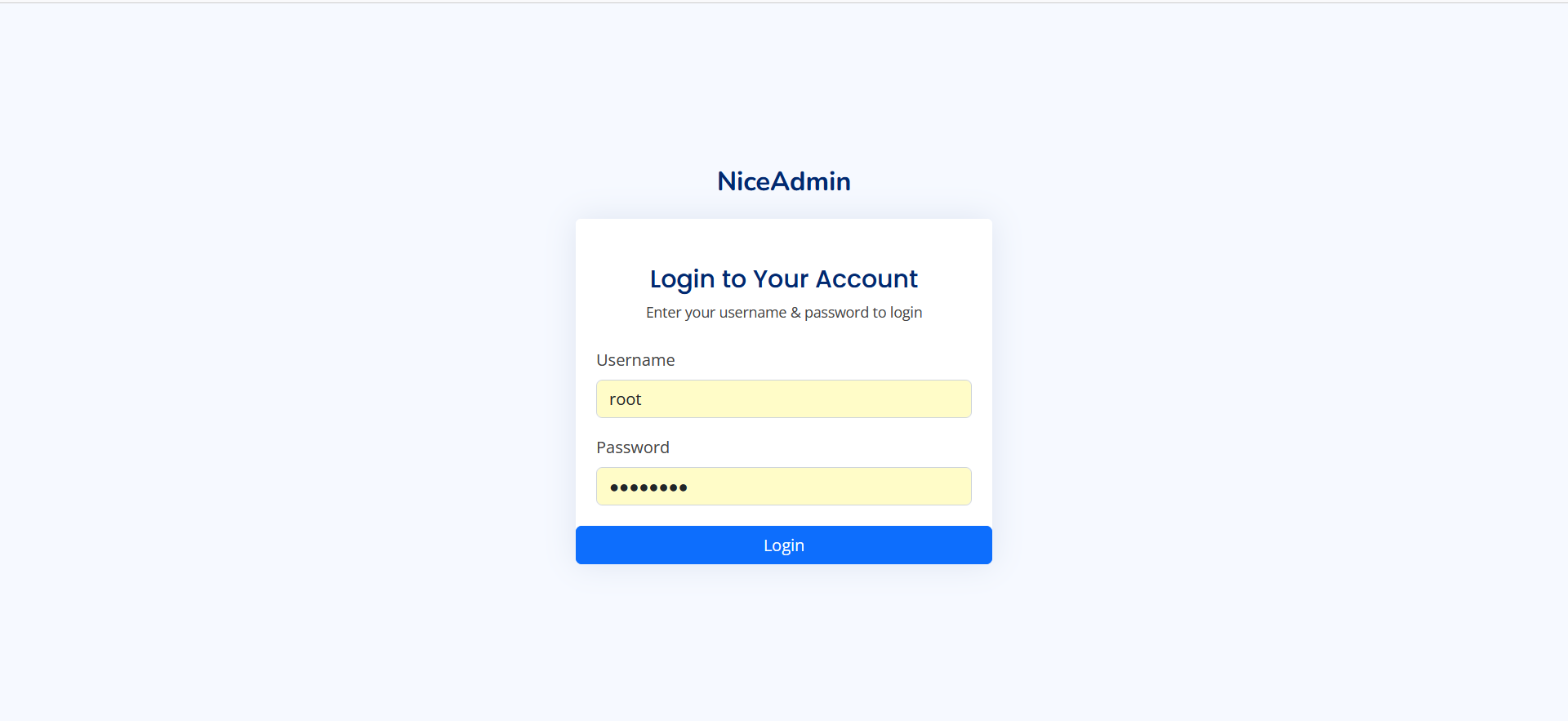
****

Рисунок - Страница авторизации для преподавателя

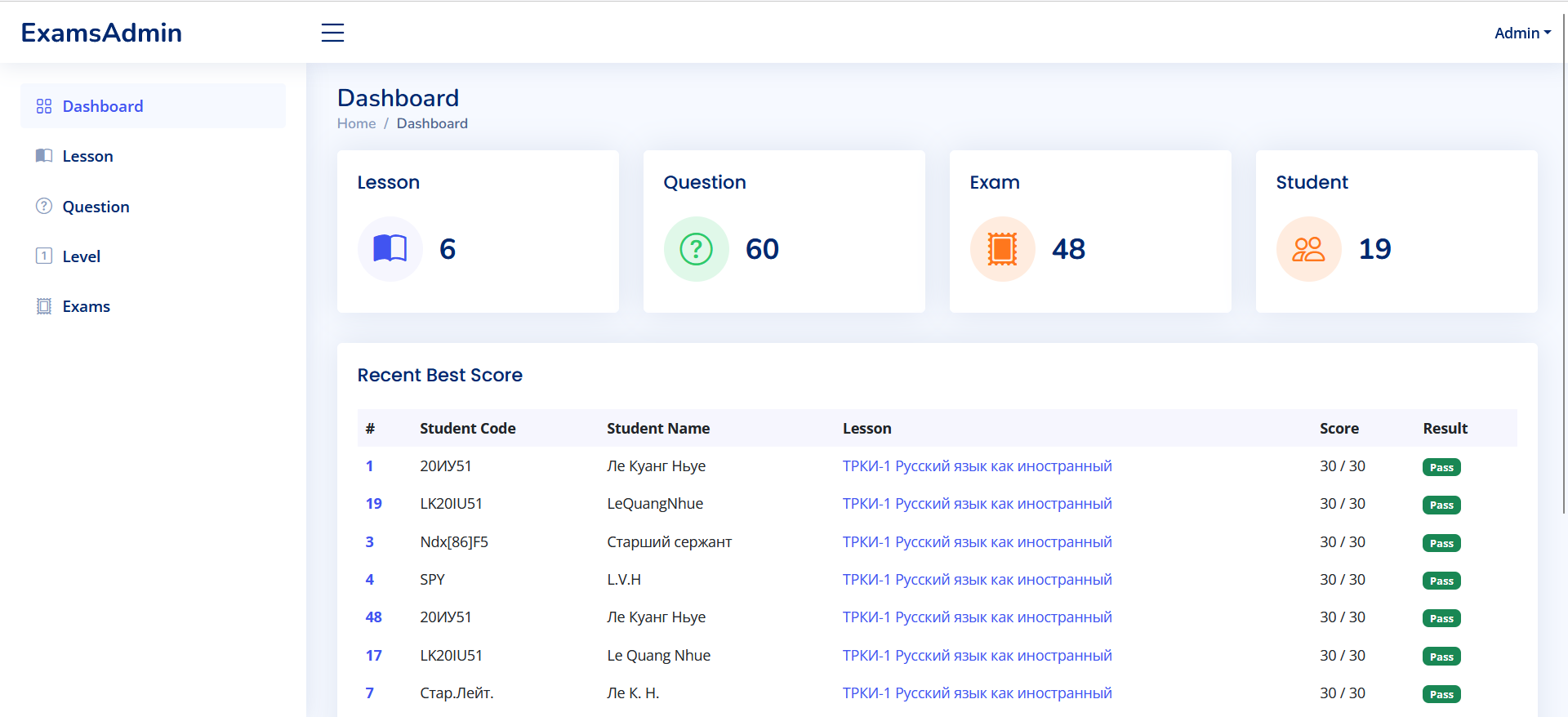
****

Рисунок - Страница приборной доски преподавателя

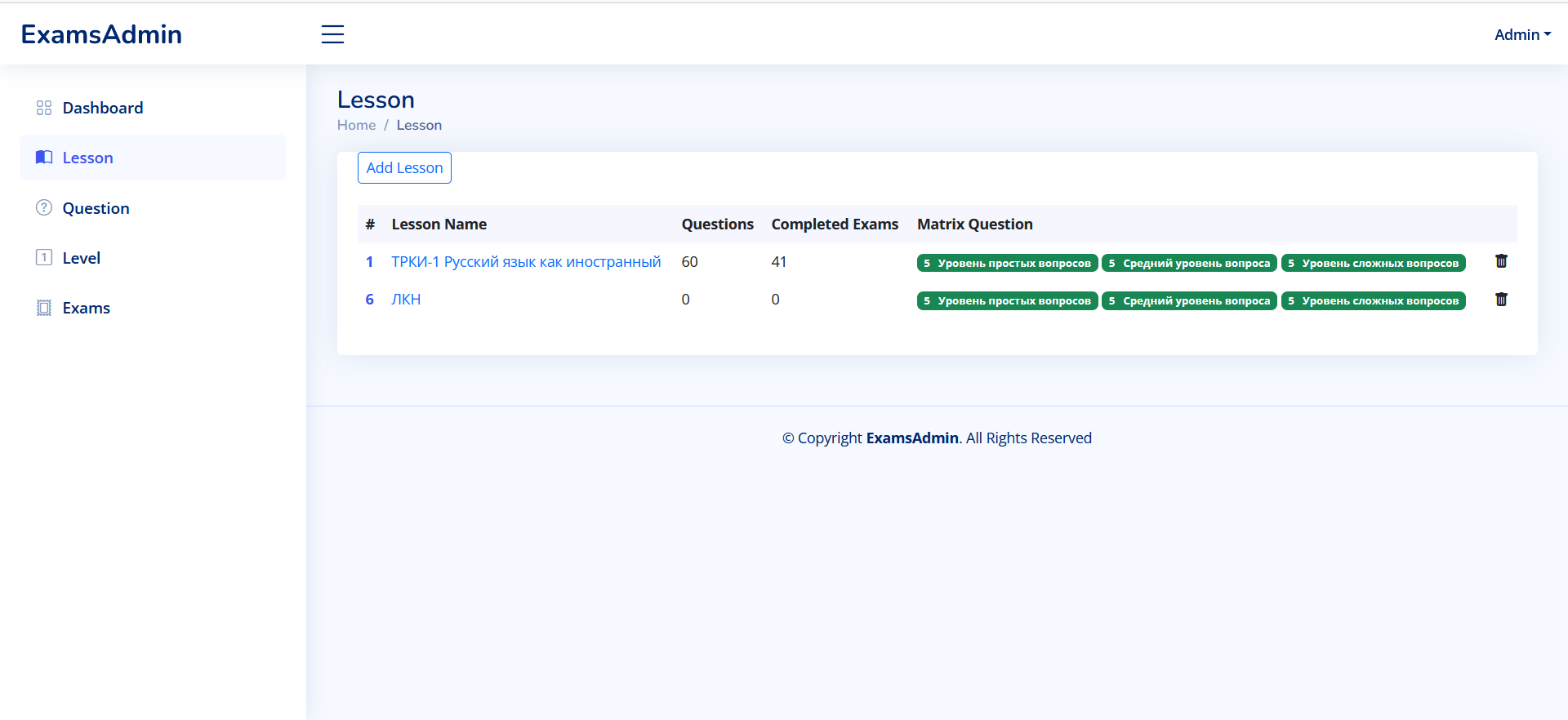
****

Рисунок - Страница перечисления предметов, созданных преподавателем

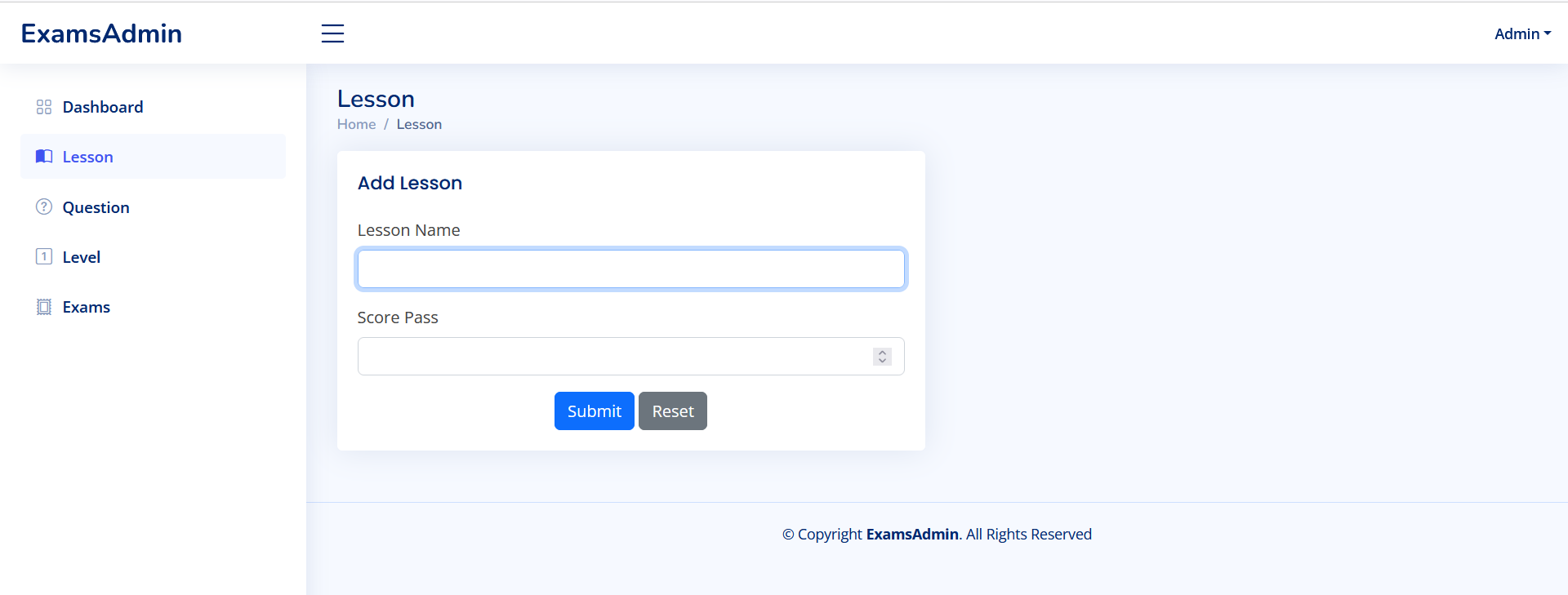
****

Рисунок - Страница добавления предмета с установкой некоторых полей

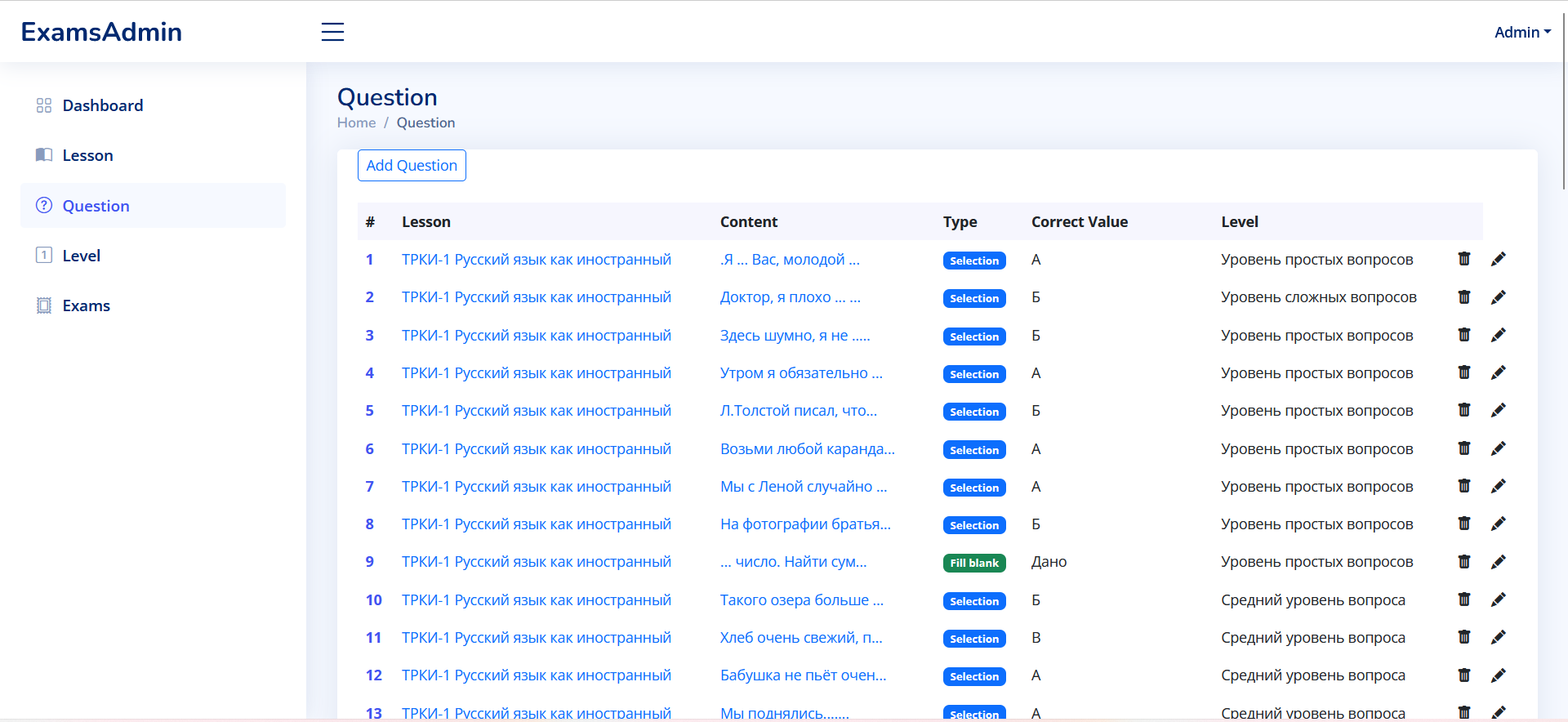
****

Рисунок – Страница перечисления вопросов, созданных преподавателем

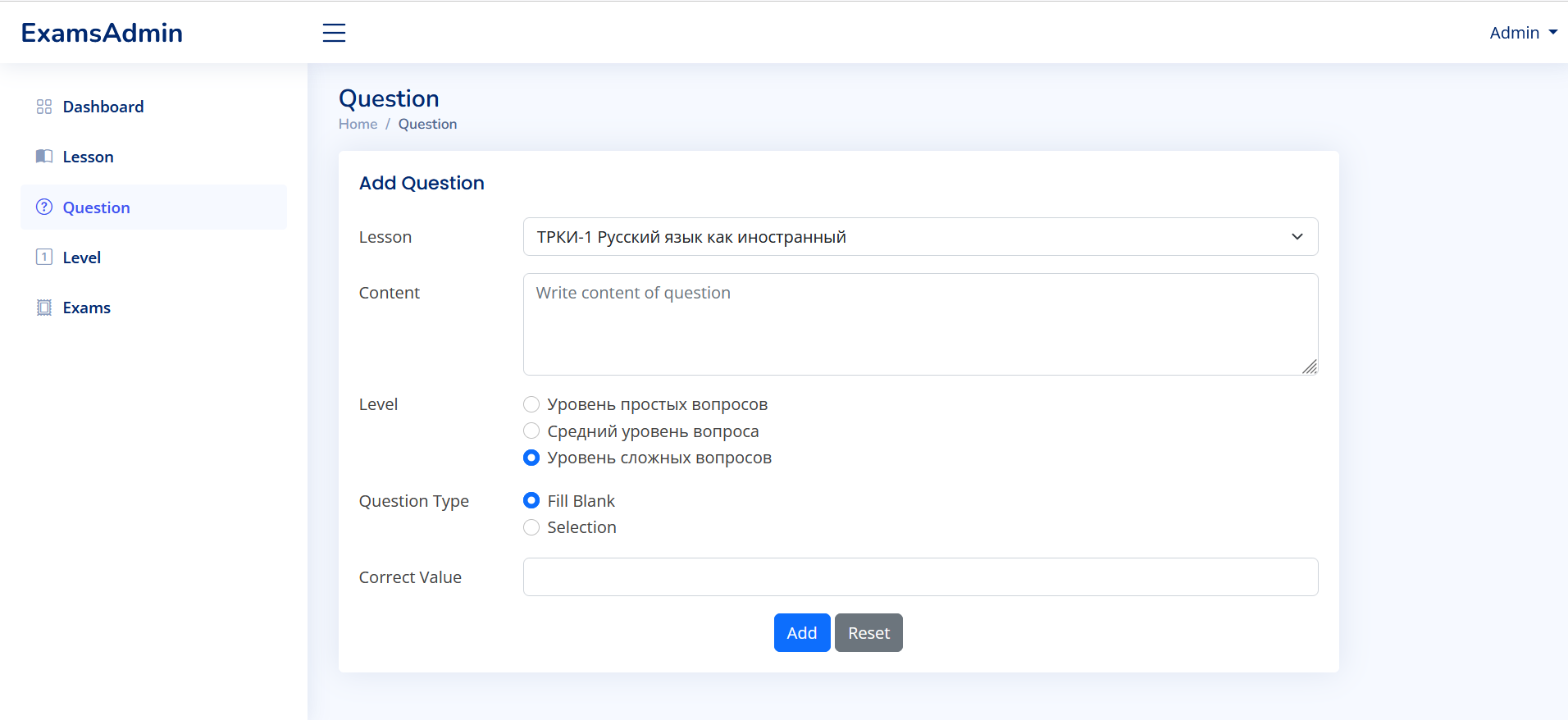
****

Рисунок - Страница добавления вопроса с установкой некоторых полей

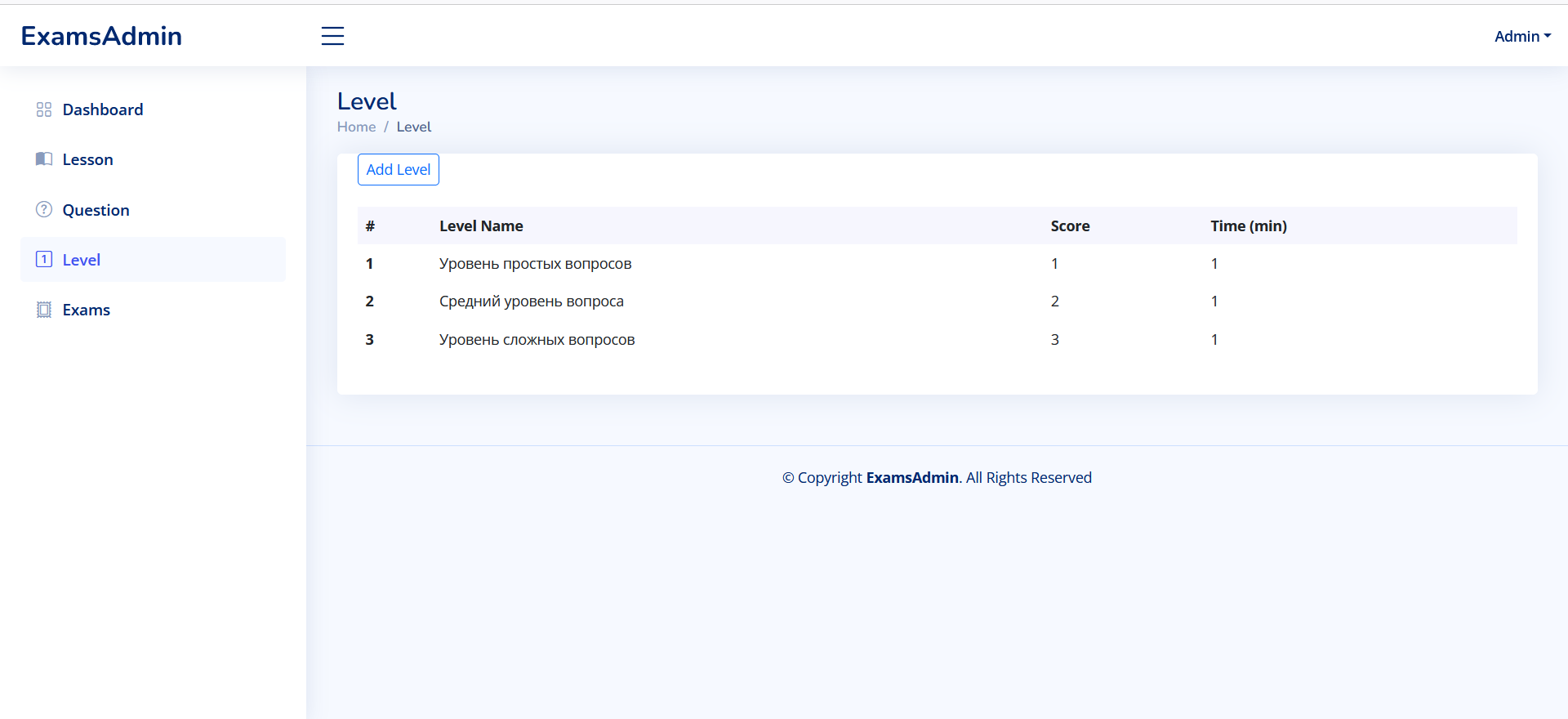
****

Рисунок - Страница перечисления уровней, созданных преподавателем

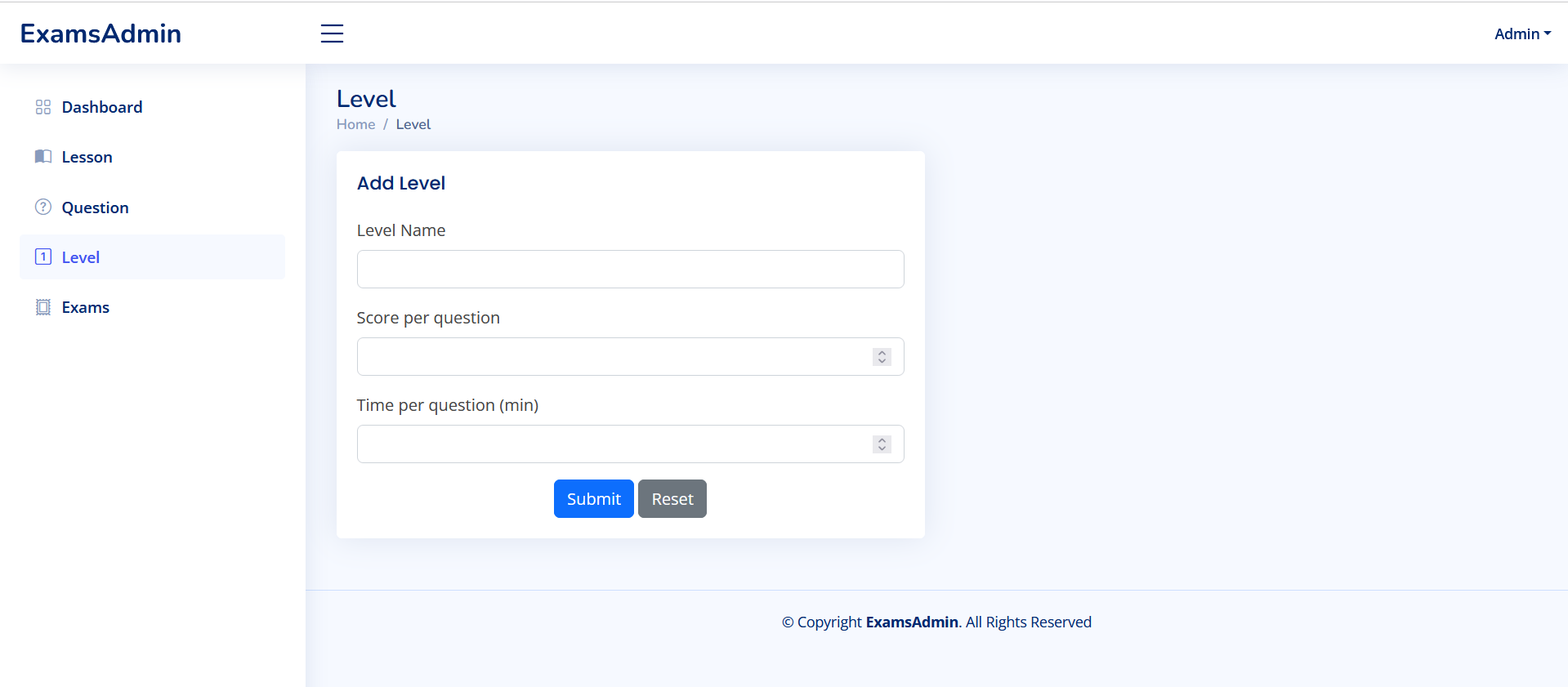
****

Рисунок - Страница добавления уровня с установкой некоторых полей

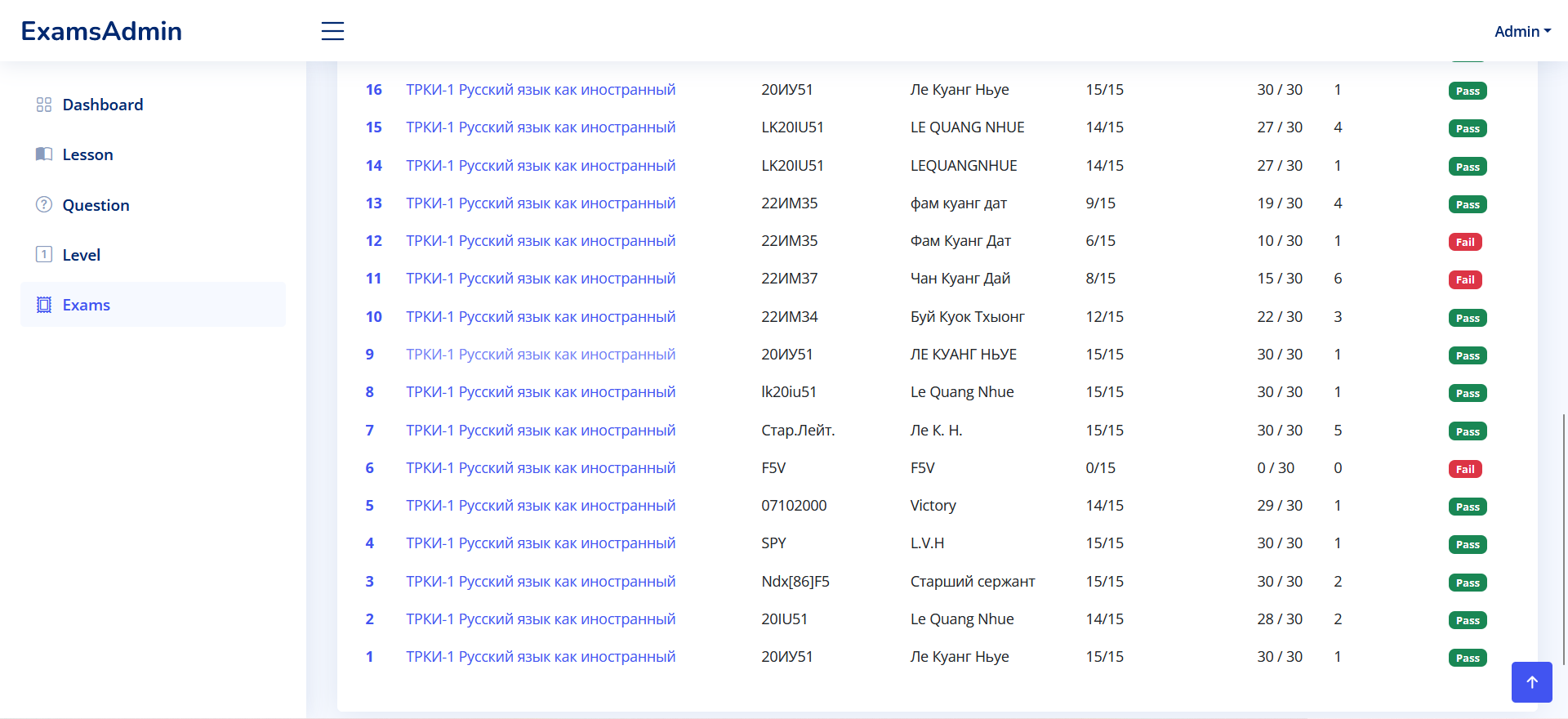
****

Рисунок - Страница свода выполненных экзаменов, в котором хранены все данные какого-то экзамена

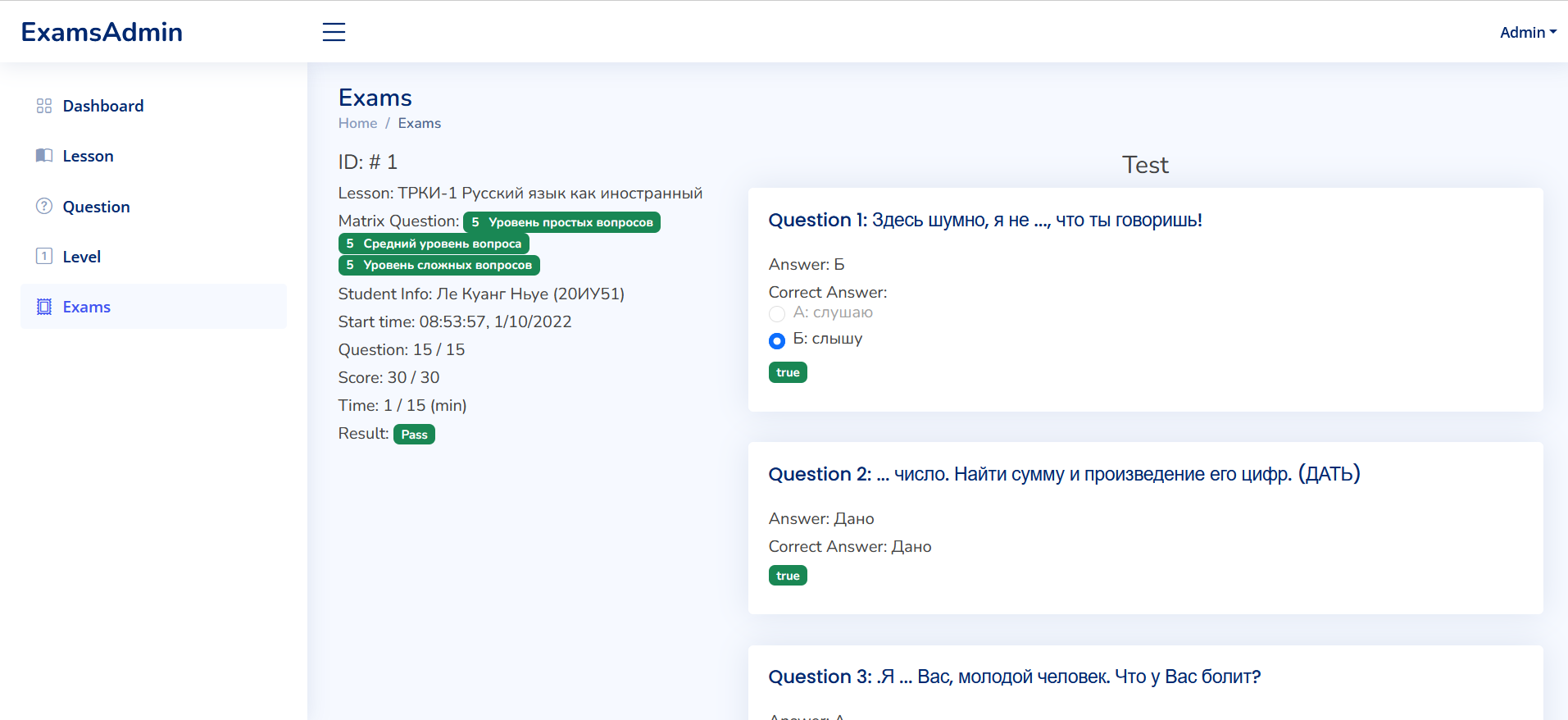
****

Рисунок - Страница деталей одного экзамена

Для студентов, все возможности работы которых на некоторых страницах и их формы интерфейсов представлены на рисунках х-у.

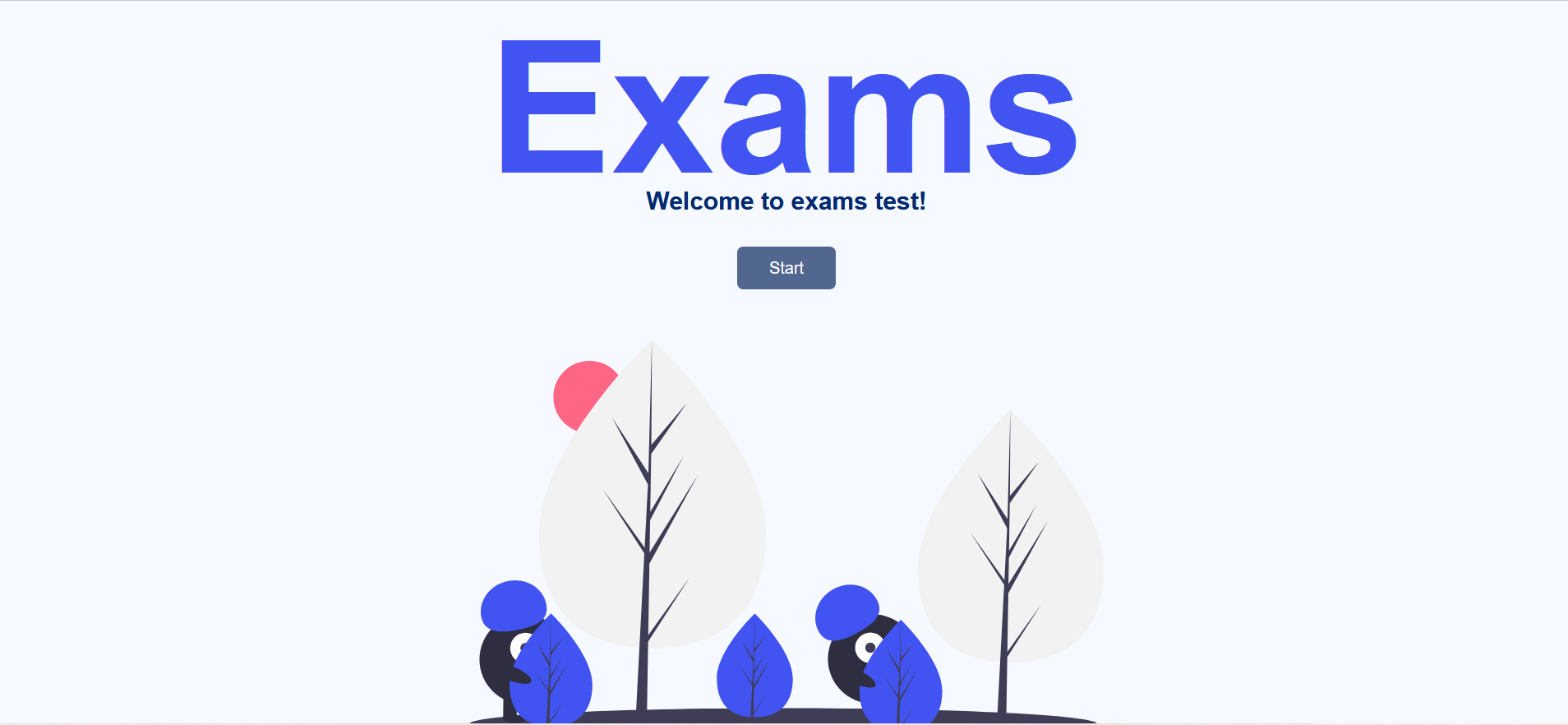


Рисунок – Начальная страница для студента

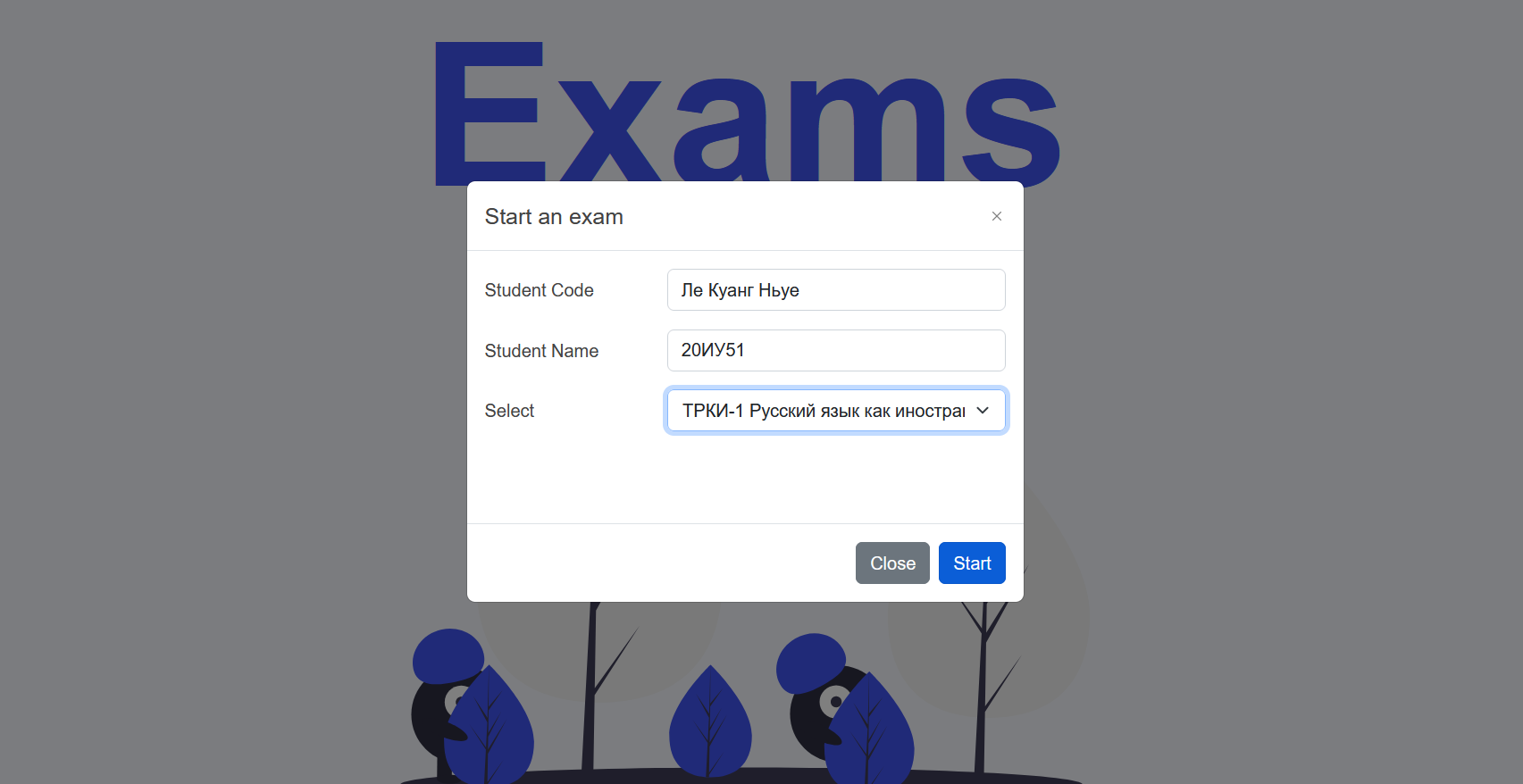
****

Рисунок – Страница окна выполнения полей для студента

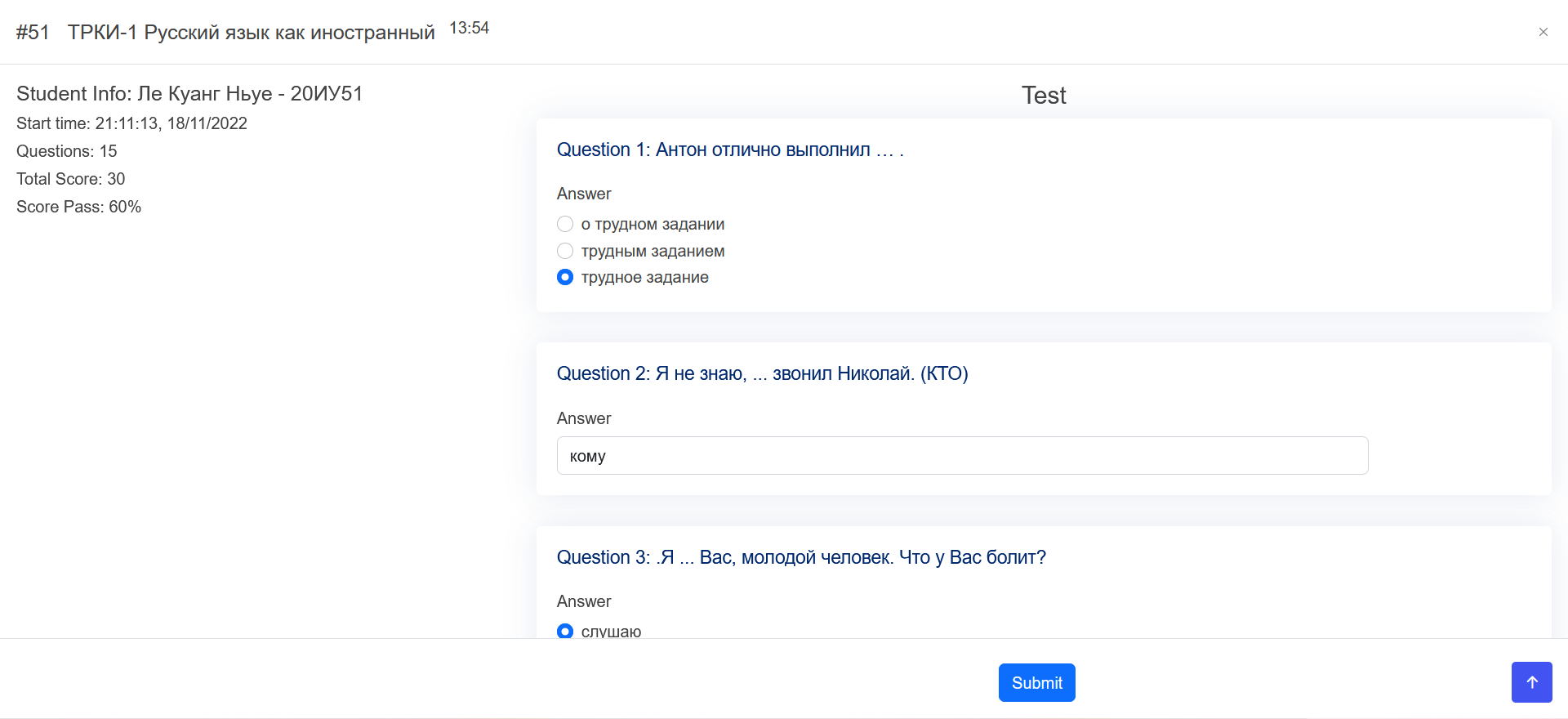
****

Рисунок – Страница выполнения экзамена для студента

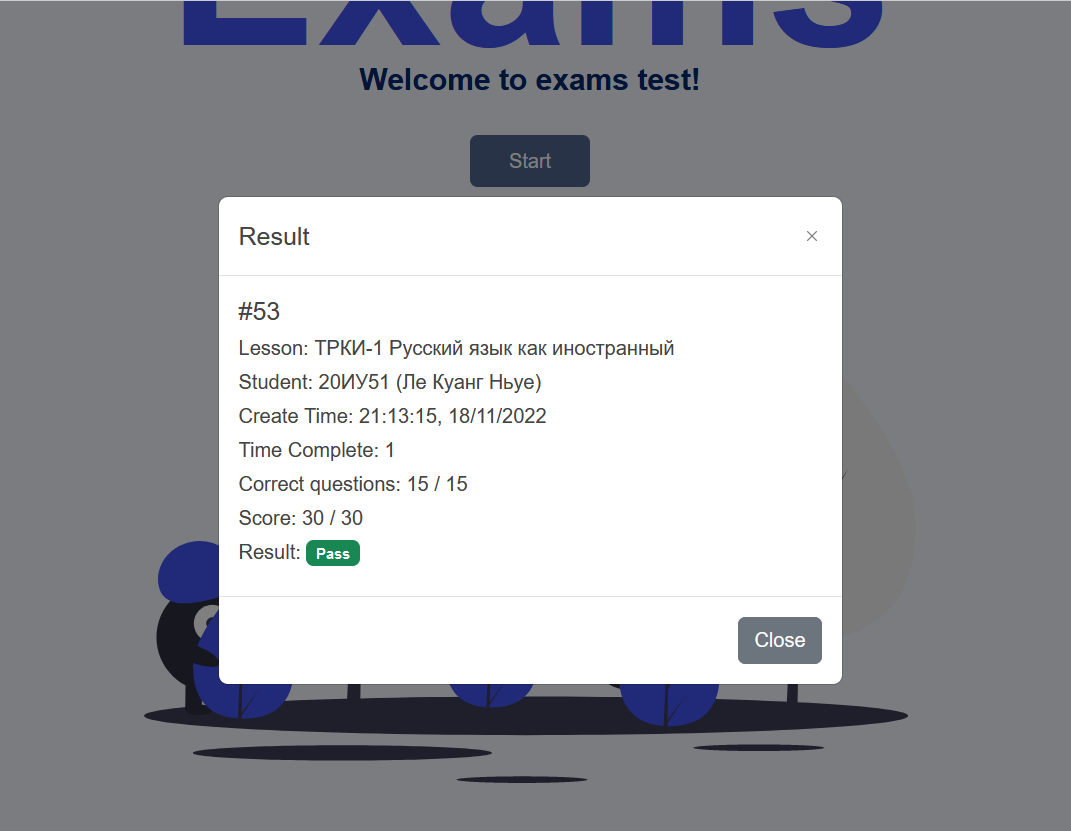
****

Рисунок – Страница смотра финального результата экзамена для студента