1. **Выбор и обоснование средств проектирования и реализации** 
   1. **Средства проектирования**

Для проектирования системы использовались следующие инструменты:

* **DBDesigner** выбран как инструмент для проектирования баз данных благодаря своему интуитивному интерфейсу, широкому набору функций, включая генерацию скриптов для различных СУБД, и статусу свободного программного обеспечения с открытым исходным кодом, что обеспечивает гибкость и доступность для разработчиков.
* **Figma** был выбран из-за его высокой гибкости и универсальности. Это интуитивно понятное средство, которое обеспечивает не только дизайн интерфейсов, но и создание детальных прототипов. А также возможность работать в браузере без необходимости установки дополнительного программного обеспечения повлияла на выбор в пользу этого инструмента.
* **StarUML** был выбран в качестве средства проектирования из-за его многофункциональности и удобства использования. Этот инструмент предоставляет широкий спектр возможностей для моделирования, включая диаграммы классов, вариантов использования и многие другие. Интерфейс StarUML интуитивно понятен, что упрощает процесс создания и редактирования диаграмм.
  1. **Средства реализации**

Выбор использования технологий React, Node.js, Express.js и PostgreSQL для реализации проекта был обоснован рядом причин.

Во-первых, React обеспечивает создание приложения с использованием компонентов, которые могут быть многократно использованы. В случае автоматизированной системы для художественной школы, это может включать компоненты для управления расписанием занятий, регистрацией студентов, отслеживания прогресса и других функций. Компоненты облегчают поддержку и модификацию системы в будущем.

Во-вторых, выбор Node.js и Express.js для серверной части приложения обусловлен их преимуществами в области разработки веб-приложений. Node.js является средой выполнения JavaScript на стороне сервера, что позволяет использовать единый язык программирования как на клиентской, так и на серверной стороне. Express.js, в свою очередь, является легковесным и гибким фреймворком для создания веб-приложений на основе Node.js. Он предоставляет удобные инструменты для маршрутизации, обработки запросов и других задач, необходимых для разработки серверной части приложения.

PostgreSQL была выбрана в качестве базы данных для хранения информации и выполнения SQL-запросов. PostgreSQL является одной из самых популярных и широко используемых реляционных баз данных. Она обладает простым в использовании интерфейсом, хорошей производительностью и поддерживает полный набор функций SQL. В контексте проекта позволяет эффективно хранить и организовывать данные.

Таким образом, выбор технологий React, Node.js, Express.js и PostgreSQL для реализации данного проекта обусловлен их преимуществами и соответствием требованиям проекта, включая гибкость, эффективность и удобство использования в контексте создания визуальной новеллы для обучения SQL-запросам.

1. **Проектирование архитектуры приложения**

Для приложения была выбрана клиент-серверная архитектура, которая представлена на рисунке 1.

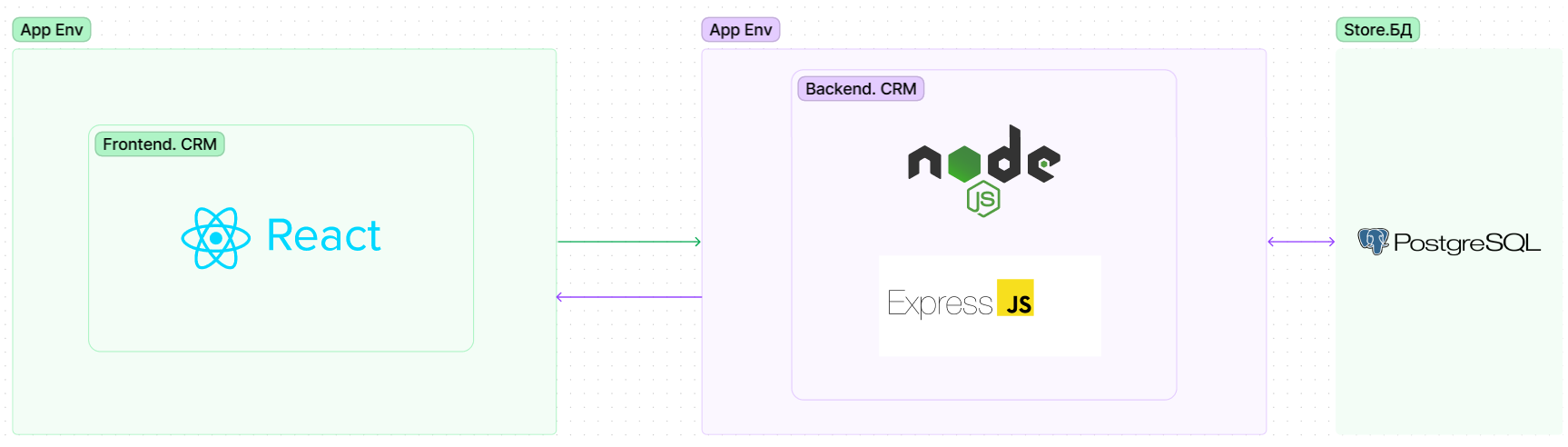
****

Рисунок 1 - Архитектура приложения

Клиент-серверная архитектура представляет собой структурный подход к организации и взаимодействию программных систем. В этой архитектуре функции приложения распределены между клиентской (пользовательской) и серверной (центральной) частями. На клиентской стороне располагается интерфейс пользователя, обеспечивающий визуальное взаимодействие. Кроме того, часть логики приложения может быть реализована на клиенте для повышения отзывчивости.

Серверная часть, в свою очередь, содержит основную бизнес-логику, обработку данных и принятие решений. Здесь также осуществляется управление данными, и их хранение может быть реализовано через работу с базой данных или другими хранилищами данных. Сервер принимает запросы от клиента, обрабатывает их и возвращает результаты выполнения запросов.

Взаимодействие между клиентом и сервером осуществляется через клиент-серверное взаимодействие по сети. Это может быть реализовано с использованием различных сетевых технологий, таких как HTTP, WebSocket и другие. Ключевое в этой архитектуре — асинхронность взаимодействия, что способствует эффективности приложения.

Данная архитектура была выбрана, потому что она обеспечивает модульность приложения, позволяя разделять бизнес-логику и пользовательский интерфейс. Это содействует масштабируемости и облегчает управление доступом к данным.

1. **Проектирование хранилища базы данных**
   1. **Основные сущности**

Система имеет следующие основные сущности (табл. 1).

Таблица 1 – Описание сущностей

|  |  |
| --- | --- |
| **Сущность** | **Атрибуты** |
| Преподаватель | ФИО преподавателя, пол, паспорт, телефон, дата рождения |
| Группа | Название группы, ID преподавателя, ID учебной программы |
| Ученик | ФИО ученика, дата рождения, телефон, почта, ID группы |
| Расписание | ID преподавателя, ID предмета, ID времени, ID дня, ID группы, ID кабинета |
| Посещаемость | ID расписания, ID ученика, присутствие, причина |
| Успеваемость | ID расписания, ID ученика, оценка |

* 1. **Логическая модель данных**

На рисунке 2 представлена логическая модель данных.

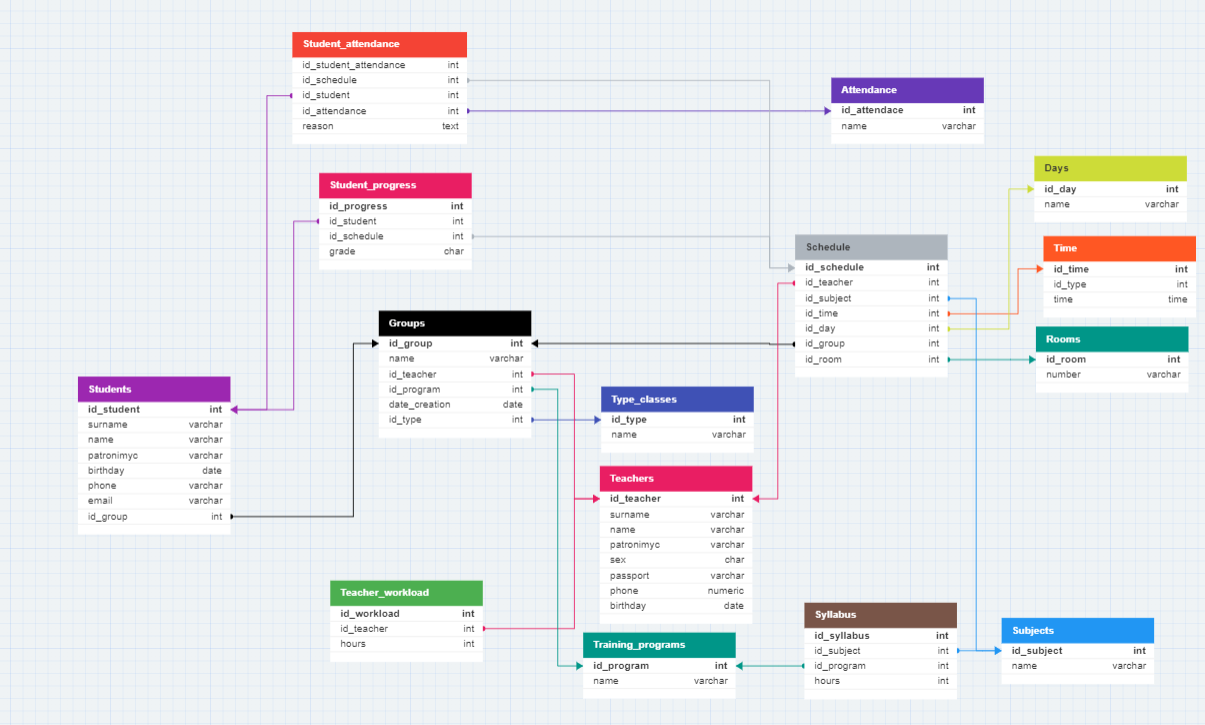


Рисунок 2 - Логическая модель данных

1. **Проектирование пользовательского интерфейса**

На рисунке 3 представлена иерархия экранов.

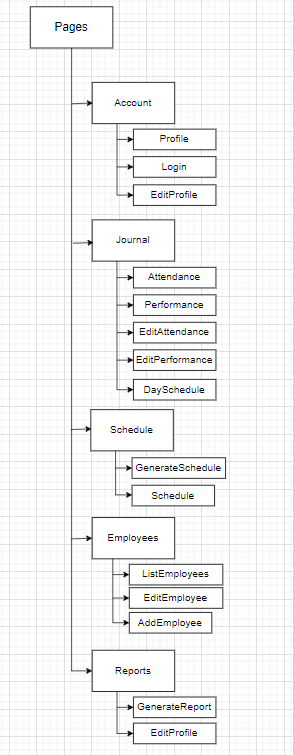


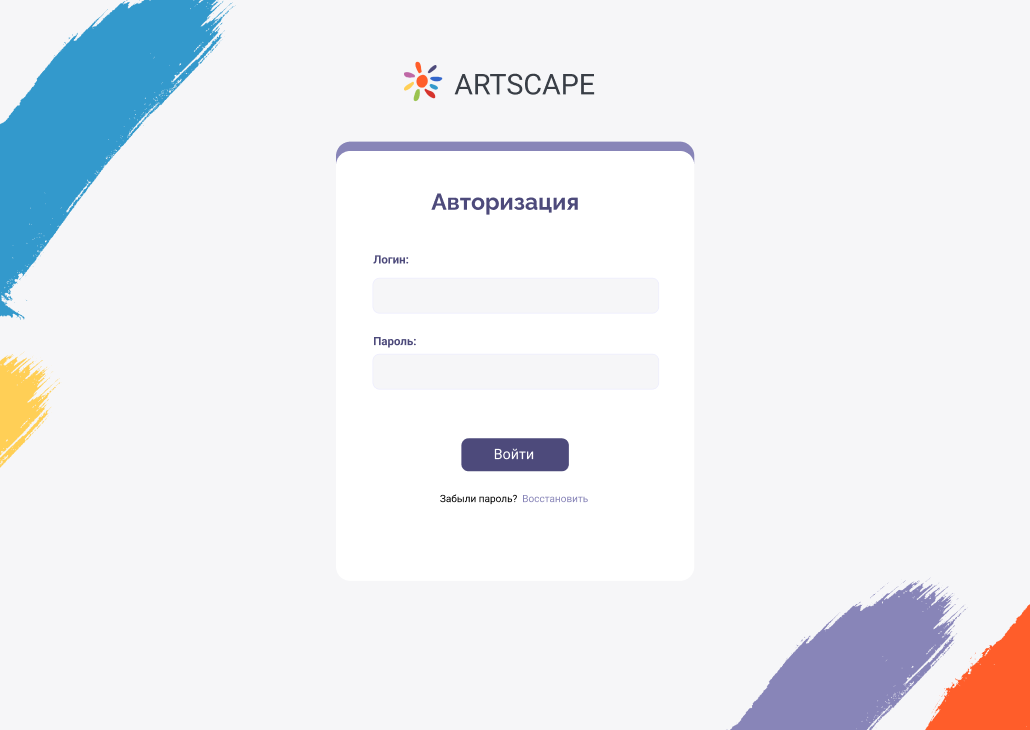
Рисунок 3 – Иерархия экранов

В таблице 2 приведено описание экранов системы.

Таблица 2 – Описание экранов системы

|  |  |  |
| --- | --- | --- |
| **Номер экрана** | **Краткое название** | **Описание экрана и его поведение** |
| **1** | **Account** |  |
| 1.1 | Profile | Личный кабинет пользователя, где представлены его персональные данные и фотография |
| 1.2 | Login | Форма авторизации, в которой необходимо ввести логин и пароль. При вводе некорректных данных пользователю выдается соответствующее сообщение. |
| 1.3 | EditProfile | Страница, на которой представлена возможность редактирования личной информации. Она содержит поля, в которых введены текущие данные. При вводе некорректных значений, например, вводе цифры в поле ввода ФИО пользователю выдается сообщение. Также, если пользователь пытается сохранить данные с пустыми полями, которые обязательны для заполнения, то система тоже выдают соответствующее сообщение |
| **2** | **Journal** |  |
| 2.1 | Attendance | На данной странице представлена таблица с учащимися соответствующей группы. В полях таблицы содержится выпадающий список со следующими значениями: «Был», «Не был», «Болеет», «Другое». Справа в нижнем углу представлена кнопка «Сохранить», при нажатии на которую происходит запись данных в БД |
| 2.2 | Performance | На данной странице представлена таблица с учащимися соответствующей группы. В полях таблицы содержится выпадающий список со следующими значениями: «5», «4», «3», «2». Также в ячейке таблицы есть иконка изображения, при нажатии на которую будет выведено модальное окно с оцениваемым рисунком. Справа в нижнем углу представлена кнопка «Сохранить», при нажатии на которую происходит запись данных в БД |
| 2.3 | DaySchedule | Страница, которая содержит расписание преподавателя на текущий день. После нажатия на определенное занятие будет совершен переход на страницу «Attendance» |
| **3** | **Schedule** |  |
| 3.1 | Schedule | Страница с расписанием на неделю. Показывает такие данные, как название предмета, аудитория, время и имя преподавателя |
| **4** | **Reports** |  |
| 4.1 | GenerateReport | Страница, содержащая поля ввода периода, на основе которого необходимо сформировать отчет. Также справа от них расположена кнопка «Показать», при нажатии на которую будет сформирован отчет в виде диаграмм и графиков |

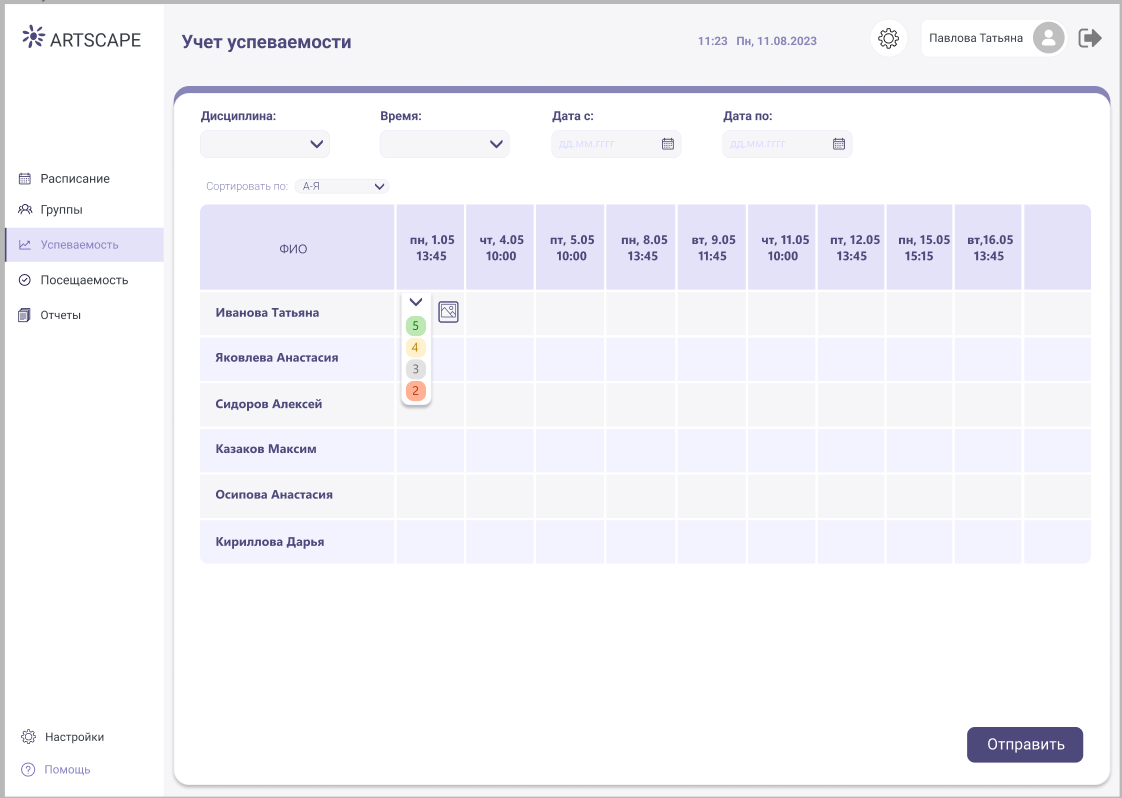
На рисунке 3.1 представлен макет страницы авторизации в системе.



**Рисунок 3.1 – Макет страницы авторизации**

Представленный на рисунке 3.2 макет экрана предназначен для заполнения документа в рамках задачи учета успеваемости с использованием учетной записи преподавателя. Навигационное меню находится слева для обеспечения удобства работы с системой. Для упрощения процесса заполнения документа в правой верхней части экрана отображается текущий день недели и время. Основная функциональность задачи «Учет успеваемости» расположен в центральной части экрана.

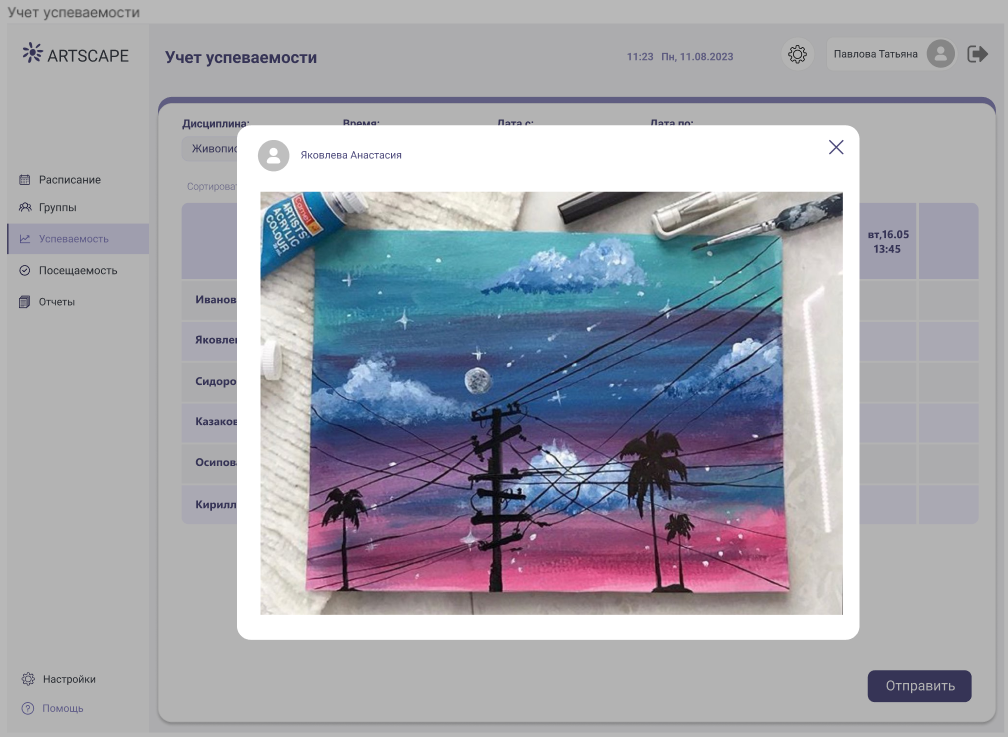
Для облегчения работы с документом, на экране выводится текущий день недели и время в правой верхней части экрана, а также используются эргономические особенности работы с документом в цветовом решении, такие как выделение информационной части документа. Возможные оценки учеников отображаются в документе при помощи цветовой индикации в зависимости от высоты оценки. Таким образом, оценка «5» будет выделяться зеленым цветом, «4» – желтым, «3» – серым и «2» – красным, как самая низкая оценка.



**Рисунок 3.2 – Макет экрана «Учет успеваемости»**

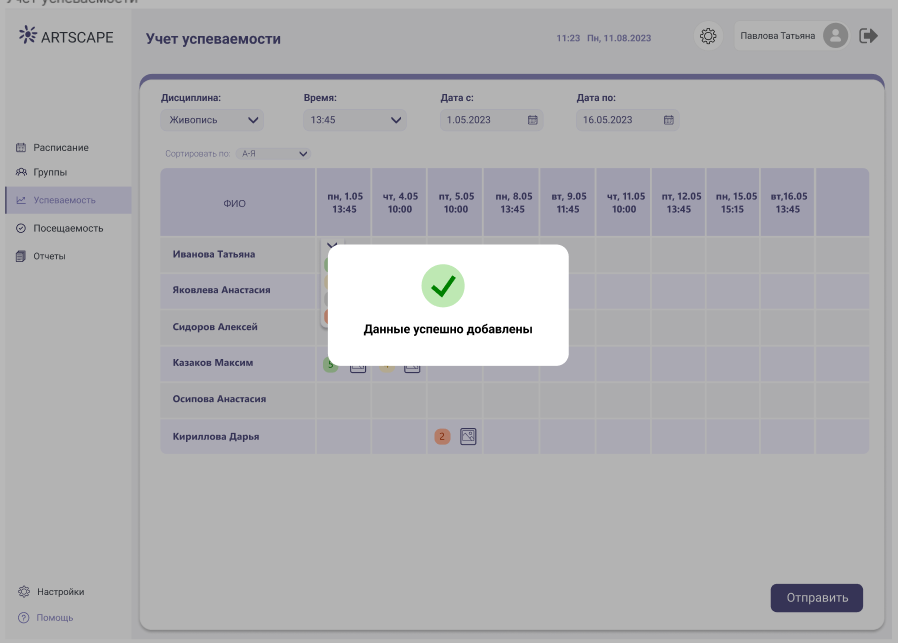
Для того чтобы предотвратить возможные ошибки пользователя при выборе дат, необходимо ввести ограничения при использовании календаря. Конкретно, при выборе даты начала периода «Дата с», необходимо заблокировать доступ к датам, которые находятся в будущем, т.е. даты, которые больше текущей даты. При выборе даты окончания периода «Дата по», необходимо заблокировать доступ к датам, которые меньше даты начала периода «Дата с» и датам, которые больше текущей даты. Таким образом, ограничения позволят пользователю выбрать только допустимые даты, что снизит вероятность возникновения ошибок при заполнении документа.

После нажатия на кнопку изображения будет выведено модальное окно с именем ученика и фотографией его работы (см. рисунок 3.3). Преподаватель может его просмотреть и проставить оценку.



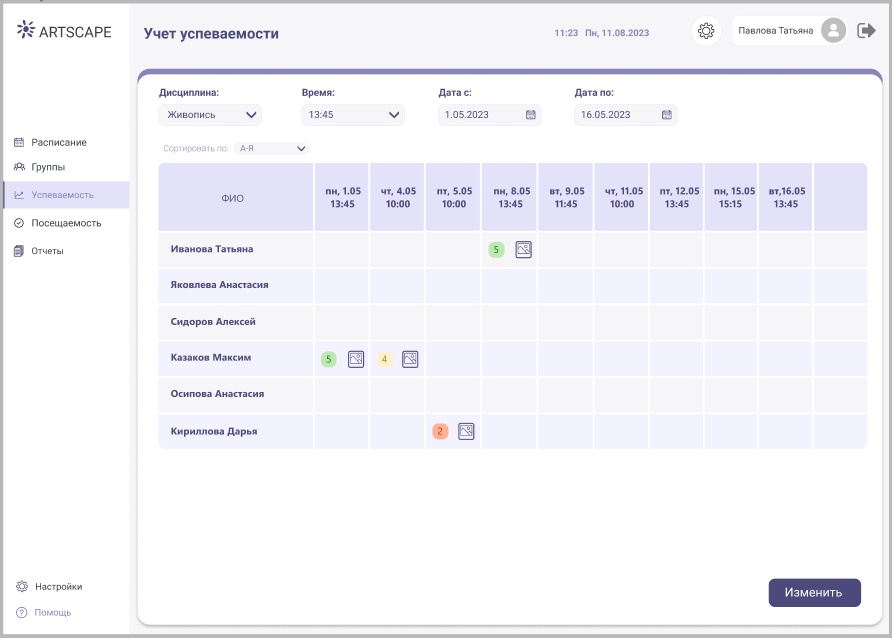
**Рисунок 3.3 – Окно с рисунком ученика**

После нажатия на кнопку «Отправить» данные будут сохранены в базе данных. Если запрос успешно выполнен, то пользователю будет выведено уведомление об успешной операции (см. рисунок 3.4) в соответствующем окне.



**Рисунок 3.4 – Успешная запись в БД**

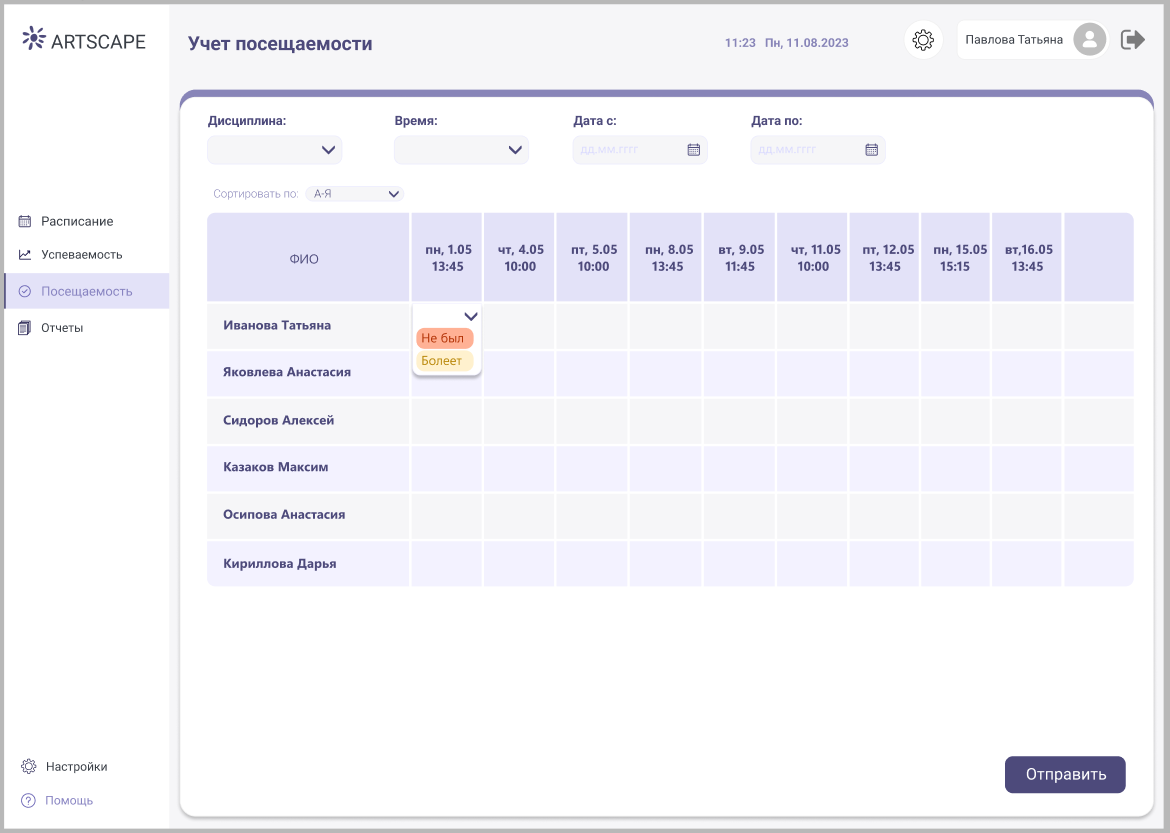
После завершения процесса добавления записи, кнопка «Добавить» преобразуется в кнопку «Изменить» (см. рисунок 3.5), и пользователю становится недоступно внесение изменений в текущую таблицу. Доступ к функции внесения данных будет восстановлен только после того, как пользователь нажмет на кнопку «Изменить», чтобы исправить ошибку, если таковая обнаружится после отправки успеваемости.



**Рисунок 3.5 – Состояние системы после закрытия окна**

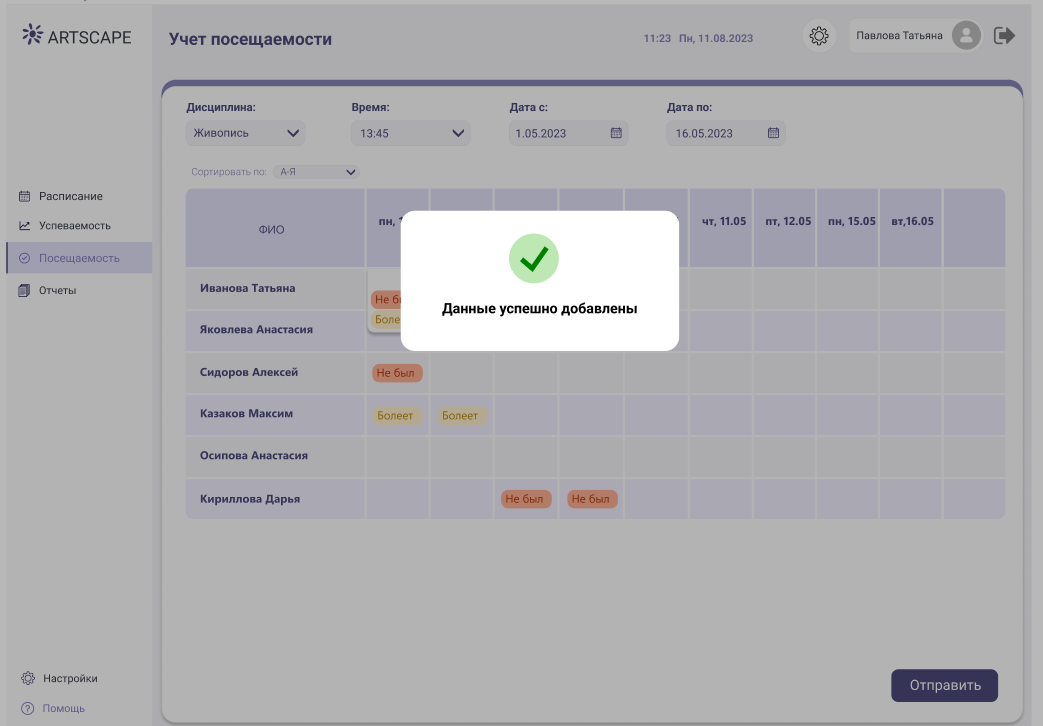
После выполнения действия «Изменить», кнопка с меткой «Изменить» возвращается в исходное состояние, т.е. метка на ней снова становится «Отправить». При этом пользователь может повторно внести необходимые изменения в таблицу и отправить их на сохранение в БД. Для этого необходимо выполнить те же действия, которые описаны выше для действия «Отправить».

На рисунке 3.6 представлен макет экрана, предназначенный для учета посещаемости с использованием учетной записи преподавателя. Навигационное меню находится слева для обеспечения удобства работы с системой. Для упрощения процесса заполнения документа в правой верхней части экрана отображается текущий день недели и время. Основная функциональность задачи «Учет посещаемости» расположен в центральной части экрана.



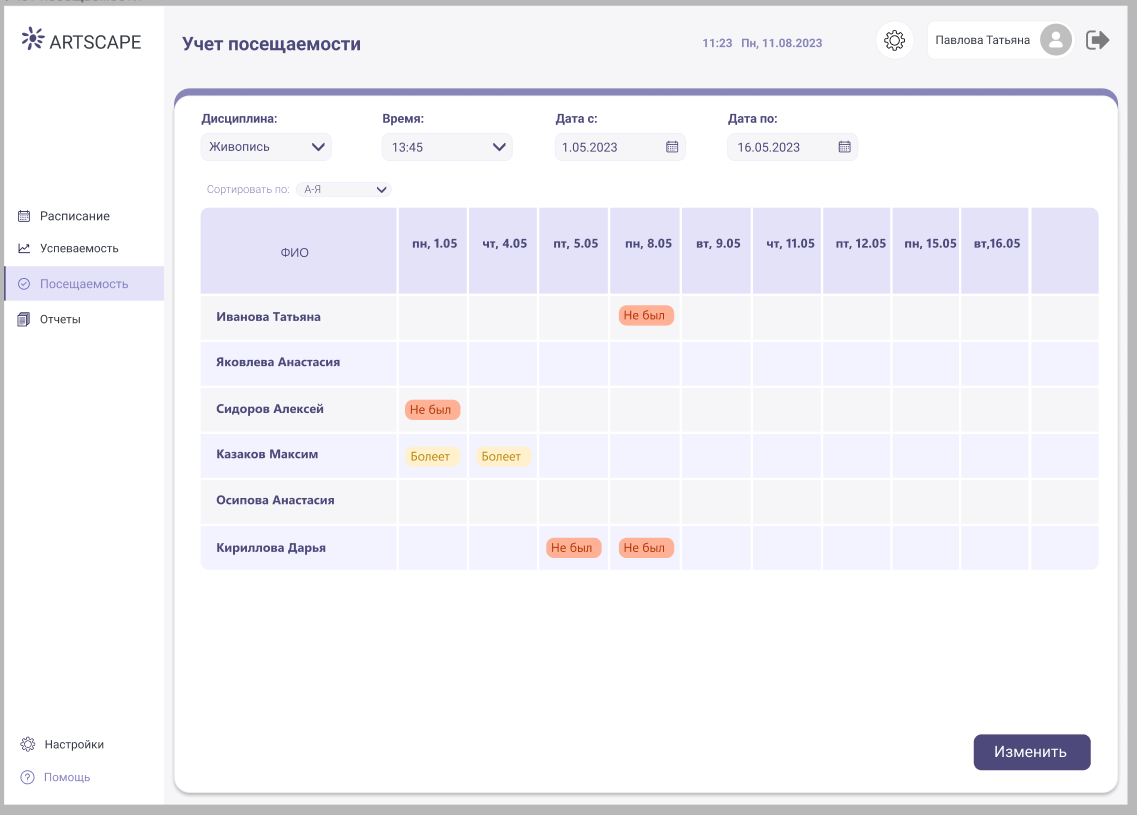
**Рисунок 3.6 – Макет экрана «Учет посещаемости»**

После нажатия на кнопку «Отправить» данные будут сохранены в базе данных. Если запрос успешно выполнен, то пользователю будет выведено уведомление об успешной операции (см. рисунок 3.7) в соответствующем окне.



**Рисунок 3.7 – Успешная запись в БД**

После завершения процесса добавления записи, кнопка «Добавить» преобразуется в кнопку «Изменить» (см. рисунок 3.8), и пользователю становится недоступно внесение изменений в текущую таблицу. Доступ к функции внесения данных будет восстановлен только после того, как пользователь нажмет на кнопку «Изменить», чтобы исправить ошибку, если таковая обнаружится после отправки посещаемости.

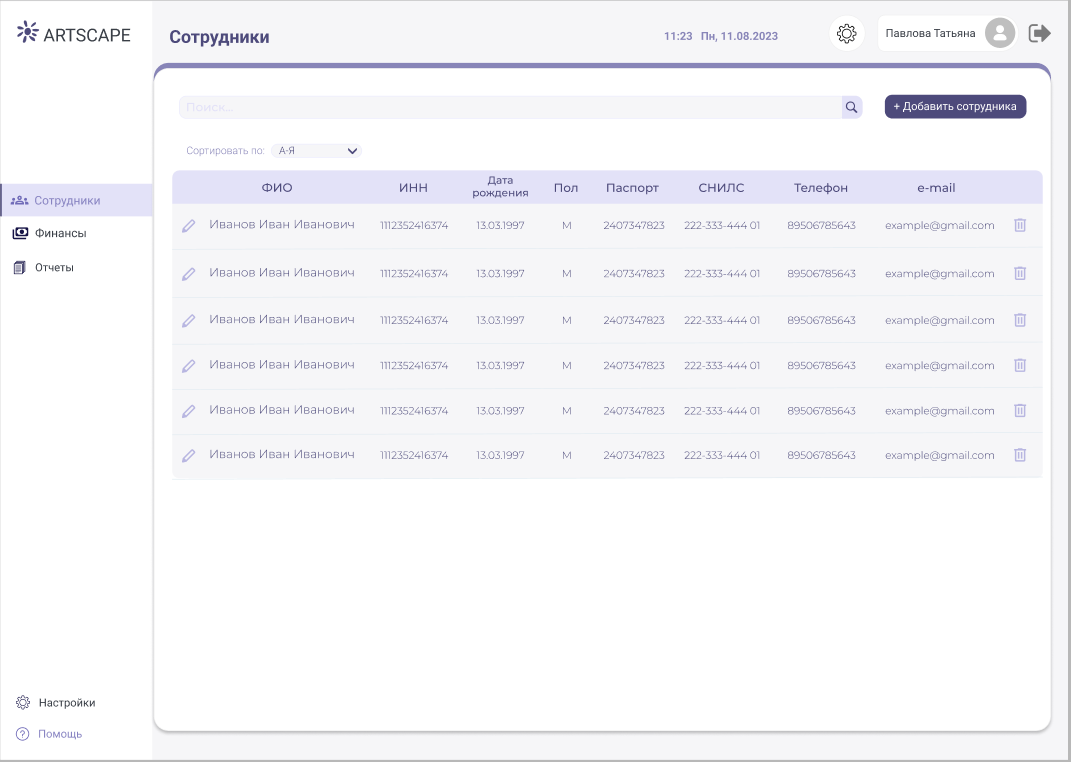


**Рисунок 3.8 – Состояние системы после закрытия окна**

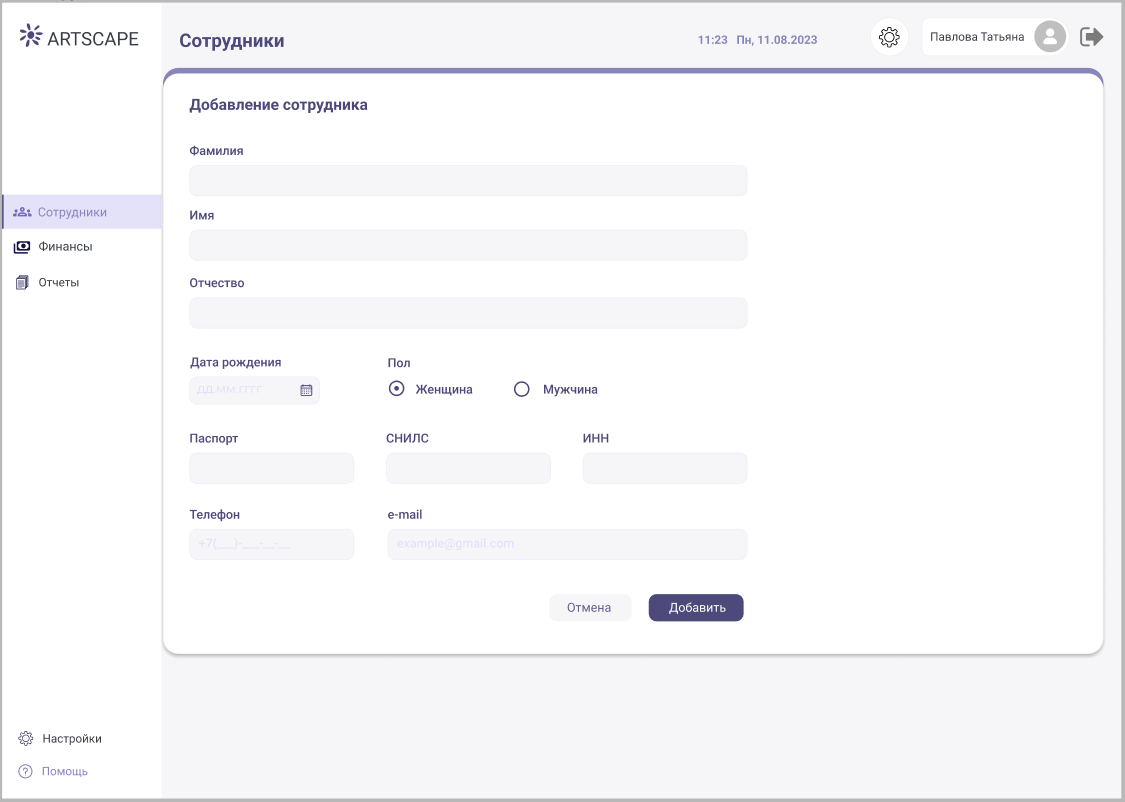
После выполнения действия «Изменить», кнопка с меткой «Изменить» возвращается в исходное состояние, т.е. метка на ней снова становится «Отправить». При этом пользователь может повторно внести необходимые изменения в таблицу и отправить их на сохранение в БД. Для этого необходимо выполнить те же действия, которые описаны выше для действия «Отправить».

На рисунке 3.9 представлен макет экрана, предназначенный для учета преподавателей со стороны директора.

При активации кнопки «Добавить сотрудника» (см. рисунок 3.10), откроется окно с формой, где необходимо внести соответствующие данные о новом преподавателе.

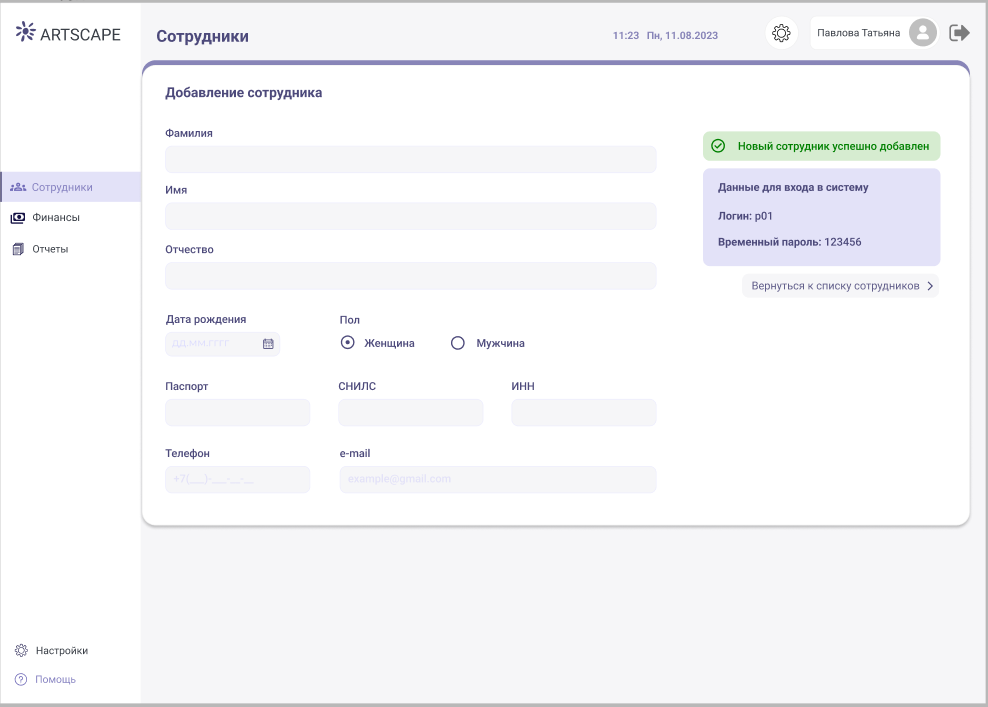


**Рисунок 3.9 – Макет страницы «Учет преподавателей»**



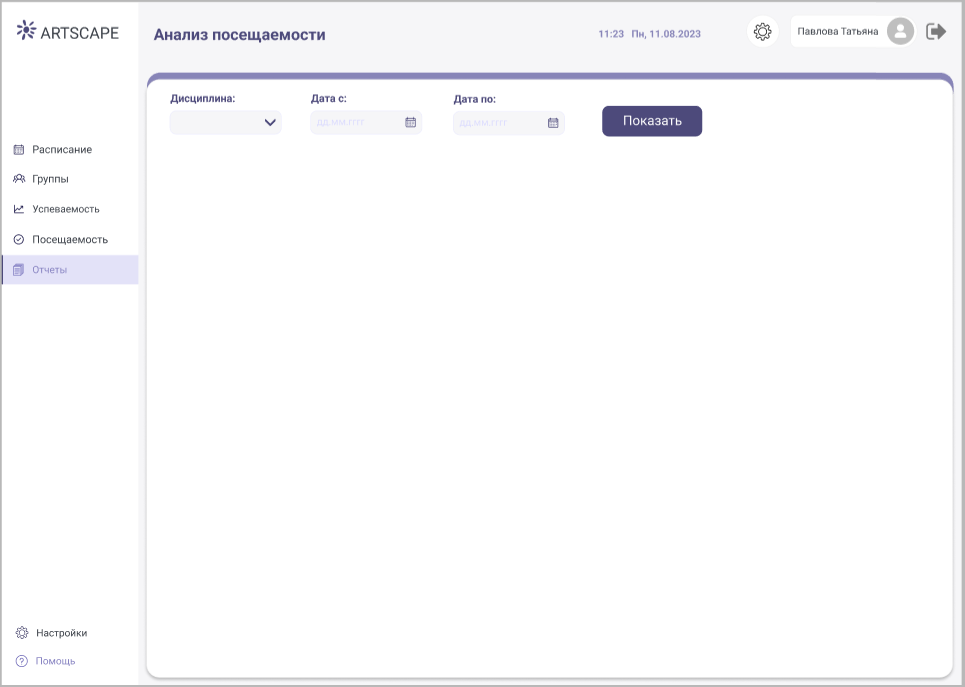
**Рисунок 3.10 – Макет экрана формы заполнения информации о преподавателе**

После заполнения формы система оповестит об успешном добавлении сотрудника, а также автоматически сгенерирует логин и временный пароль для нового пользователя (см. рисунок 3.11).



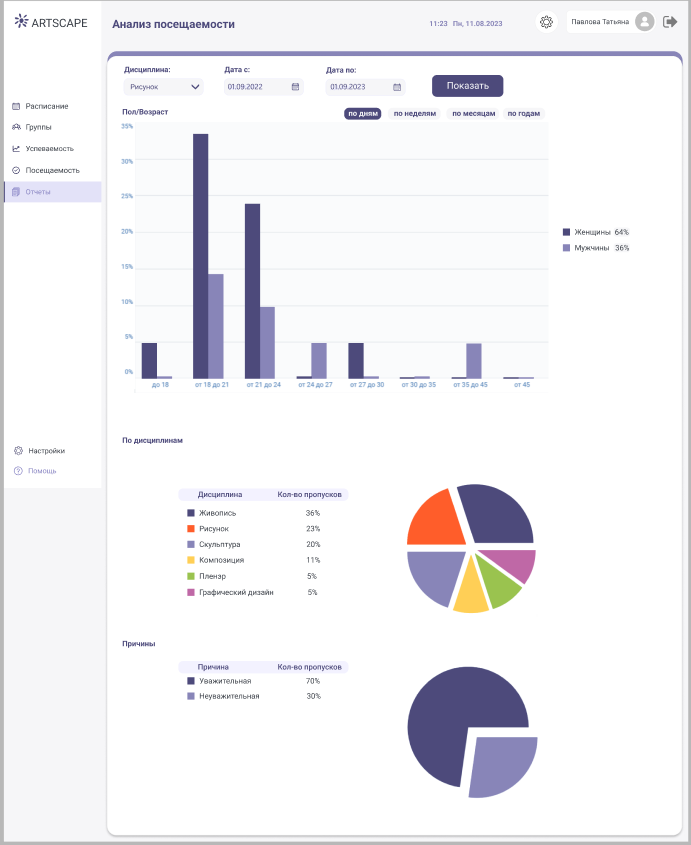
**Рисунок 3.11 – Макет экрана после успешного добавления преподавателя**

Для выполнения задачи «Анализ посещаемости» требуется ввести определенные данные, необходимые для формирования отчета. В макете экрана, который представлен на рисунке 3.12, представлены поля для ввода этой информации.

****

**Рисунок 3.12 – Макет экрана «Анализ посещаемости»**

После нажатия на кнопку «Показать» система формирует отчет (см. рисунок 3.13) согласно заданным параметрам и отображает его в графическом виде, обеспечивая более удобный способ анализа полученных данных.



**Рисунок 3.13 – Сформированный отчет**

На рисунке 6 представлена карта экранов приложения.

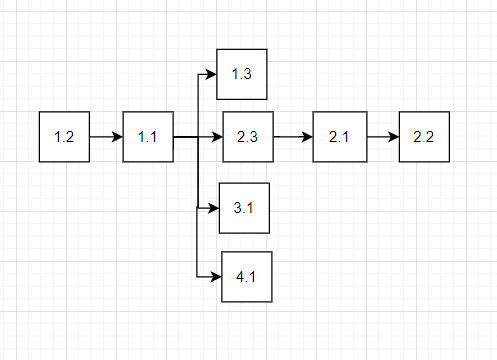


Рисунок 6 – Карта экранов