

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное учреждение высшего образования

**«Национальный исследовательский Нижегородский государственный университет им.  
Н. И. Лобачевского» (ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

Направление подготовки: «Программная инженерия»

## **ОТЧЕТ**

о прохождении ознакомительной практики

на тему:

**«Разработка веб-приложения с функциональностью CRUD:  
информационная система для музыкальной студии»**

**Выполнил:**

студент группы 3823Б1ПР2

\_\_\_\_\_ В. С. Клименко

(Подпись)

**Научный руководитель:**

кандидат ф.м.н., доцент

\_\_\_\_\_ Д. Е. Шапошников

(Подпись)

Нижний Новгород

2025

# **Оглавление**

Введение.....	3
1 Теоретическая часть (материалы и методы) .....	4
1.1 Назначение и архитектура веб-приложения .....	4
1.2 Клиентская часть веб-приложения .....	4
1.3 Серверная часть и REST API .....	4
1.4 Работа с базой данных .....	5
1.5 Механизм бронирования.....	5
1.6 Методы обработки дат и времени.....	6
1.7 Обеспечение корректности и надёжности работы .....	6
1.8 Вывод по теоретической части .....	6
2 Практическая часть (результаты и обсуждение) .....	7
2.1 Анализ требований и постановка задачи .....	7
2.2 Реализация структуры пользовательского интерфейса .....	7
2.3 Реализация динамической загрузки данных.....	7
2.4 Отображение информации об инструменте.....	8
2.5 Отображение информации о помещении.....	8
2.6 Реализация выбора дат и расчёта стоимости.....	8
2.7 Реализация механизма бронирования .....	9
2.8 Подтверждение бронирования и сохранение данных.....	9
2.9 Работа с пользовательскими данными и хранилищем браузера.....	9
2.10 Обработка ошибок и тестирование.....	9
2.11 Вывод по практической части .....	10
Заключение .....	11
Список литературы .....	12

## **Введение**

В условиях цифровизации сервисов аренды и бронирования всё большее значение приобретают веб-приложения, обеспечивающие удобный доступ пользователей к услугам в режиме реального времени. Особенно актуальной является автоматизация процессов аренды музыкального оборудования и помещений студии звукозаписи, где важны наглядность каталога, прозрачность стоимости и корректная фиксация бронирований.

Целью данной работы является разработка веб-приложения для аренды музыкальных инструментов и помещений, предоставляющего пользователю возможность просматривать каталоги, получать подробную информацию о необходимой ему категории, выбирать сроки аренды и оформлять бронирование с последующим сохранением данных в базе данных.

Разрабатываемое приложение ориентировано на пользователей музыкальной студии и частных клиентов, заинтересованных в краткосрочной и долгосрочной аренде инструментов. Система поддерживает авторизацию пользователей, расчёт стоимости аренды, подтверждение заказов и хранение истории бронирований.

В процессе разработки необходимо реализовать клиентскую часть, а также обеспечить взаимодействие с сервером посредством разработки серверной части. Серверная часть должна быть реализована с использованием базы данных для хранения информации об инструментах, помещениях и бронированиях. Такой подход обеспечивает расширяемость, модульность и удобство сопровождения системы.

# **1 Теоретическая часть (материалы и методы)**

## **1.1 Назначение и архитектура веб-приложения**

Разрабатываемое веб-приложение предназначено для автоматизации процесса аренды музыкальных инструментов и студийных помещений и управления бронированиями. Приложение реализовано в виде клиент-серверной системы, что позволяет разделить ответственность между пользовательским интерфейсом и серверной логикой, а также обеспечить масштабируемость.

Архитектура приложения построена по принципу клиент–серверного взаимодействия. Клиентская часть отвечает за отображение данных, взаимодействие с пользователем и предварительную обработку информации. Серверная часть предоставляет REST API, осуществляет бизнес-логику и обеспечивает доступ к базе данных.

Такой подход позволяет централизованно управлять данными, обеспечить их целостность и реализовать единый механизм обработки запросов от различных клиентов.

## **1.2 Клиентская часть веб-приложения**

Клиентская часть приложения реализована с использованием стандартных веб-технологий: HTML, CSS и JavaScript. HTML используется для формирования структуры страниц, CSS — для визуального оформления и адаптивного отображения элементов интерфейса, JavaScript — для реализации интерактивной логики и работы с данными.

JavaScript применяется для:

- обработки событий пользовательского интерфейса;
- динамического обновления содержимого страниц без перезагрузки;
- загрузки данных с сервера с использованием асинхронных HTTP-запросов;
- расчёта стоимости аренды на основе выбранных дат;
- валидации пользовательских данных.

Для взаимодействия с сервером используется встроенный API `fetch`, позволяющий отправлять HTTP-запросы и обрабатывать ответы в формате JSON. Асинхронный подход обеспечивает отзывчивость интерфейса и улучшает пользовательский опыт.

## **1.3 Серверная часть и REST API**

Серверная часть приложения реализована на платформе ASP.NET и предоставляет REST API для доступа к данным. Использование архитектурного стиля REST позволяет

организовать взаимодействие между клиентом и сервером с применением стандартных HTTP-методов (GET, POST, PUT, DELETE).

Основные функции серверной части включают:

- предоставление данных о музыкальных инструментах;
- предоставление данных о помещениях;
- обработку запросов на создание бронирований;
- валидацию входных данных;
- сохранение информации в базе данных.

Передача данных между клиентом и сервером осуществляется в формате JSON, что обеспечивает совместимость и удобство обработки информации на обеих сторонах.

## 1.4 Работа с базой данных

Для хранения информации используется реляционная база данных, в которой размещаются данные об инструментах, пользователях и бронированиях. База данных развёрнута в СУБД PostgreSQL. Каждое бронирование содержит сведения о пользователе, выбранном инструменте или помещении, сроках аренды и текущем статусе.

Связь между сущностями реализуется с использованием первичных и внешних ключей, что обеспечивает целостность данных и предотвращает появление некорректных записей. Доступ к базе данных осуществляется через серверную часть приложения, что исключает прямое взаимодействие клиента с хранилищем данных и повышает уровень безопасности.

## 1.5 Механизм бронирования

Процесс бронирования реализован в несколько этапов. На первом этапе пользователь выбирает инструмент/помещение и задаёт период аренды. На клиентской стороне производится расчёт итоговой стоимости на основе количества дней аренды и стоимости одного дня (для помещений расчёт производится на основе количества часов аренды и стоимости одного часа).

На следующем этапе пользователь подтверждает бронирование. После подтверждения данные отправляются на сервер посредством HTTP-запроса методом POST. Сервер обрабатывает запрос, сохраняет информацию о бронировании в базе данных и возвращает результат выполнения операции.

Для временного хранения данных бронирования на клиентской стороне используются механизмы localStorage и sessionStorage, что позволяет сохранять состояние между страницами без необходимости повторного обращения к серверу.

## **1.6 Методы обработки дат и времени**

В приложении используются методы работы с датами для корректного определения сроков аренды и расчёта стоимости. Для бронирования инструментов пользователь выбирает даты начала и окончания аренды, после чего вычисляется количество дней аренды. В случае если пользователь решит бронировать помещение, ему необходимо выбрать день, время и количество часов, на которое оно будет забронировано.

Для обеспечения согласованности данных применяется преобразование локального времени пользователя в единый формат (ISO 8601) с учётом временной зоны. Это позволяет корректно сохранять и обрабатывать временные данные на серверной стороне.

## **1.7 Обеспечение корректности и надёжности работы**

Для повышения надёжности приложения реализована обработка ошибок, возникающих при загрузке данных или выполнении сетевых запросов. В случае возникновения ошибок пользователю отображаются соответствующие уведомления в консоли Web API, а также в консоли разработчика в браузере (тестирование работоспособности веб-приложения и корректного отображения страниц происходило в Яндекс Браузере).

Также предусмотрена проверка авторизации пользователя перед подтверждением бронирования, что предотвращает создание заказов неавторизованными пользователями. Если неавторизованный пользователь делает попытку бронирования, система выдаёт сообщение о необходимости входа и переходит на соответствующую страницу.

## **1.8 Вывод по теоретической части**

В ходе изучения теоретических основ были рассмотрены принципы построения клиент-серверных веб-приложений, методы организации REST API, а также способы работы с базами данных и пользовательским интерфейсом. Полученные знания были использованы при разработке веб-приложения аренды музыкальных инструментов и студийных помещений и легли в основу практической реализации проекта.

## **2 Практическая часть (результаты и обсуждение)**

### **2.1 Анализ требований и постановка задачи**

На начальном этапе практики был проведён анализ предметной области и требований к разрабатываемому веб-приложению. Основной задачей являлось создание пользовательского интерфейса для аренды с возможностью просмотра каталога, получения подробной информации об инструменте или студийном помещении, выбора сроков аренды и оформления бронирования.

Также были определены основные функциональные требования:

- отображение списка доступных инструментов;
- отображение списка доступных помещений;
- загрузка данных с сервера через API;
- расчёт стоимости аренды в зависимости от выбранного периода;
- сохранение информации о бронировании в базе данных;
- поддержка авторизации пользователя.

### **2.2 Реализация структуры пользовательского интерфейса**

На следующем этапе была разработана структура веб-страниц приложения с использованием HTML. Были созданы основные страницы: страницы входа и регистрации, каталог инструментов, страница детального просмотра инструмента, каталог помещений, страница детального просмотра помещения, главная, контакты, профиль, абонементы, цены, страница подтверждения заказа и страница просмотра пользовательских бронирований.

При разработке интерфейса особое внимание уделялось логичной навигации и удобству использования. Все основные элементы управления (кнопки, поля ввода, списки дат) были размещены таким образом, чтобы пользователь мог последовательно выполнять действия по бронированию без затруднений.

### **2.3 Реализация динамической загрузки данных**

После создания статической структуры страниц была реализована динамическая загрузка данных об инструментах и помещениях с серверной части приложения. Для этого использовались асинхронные HTTP-запросы с применением API fetch.

При загрузке страницы детального просмотра инструмента/помещения выполнялся запрос к серверу для получения информации по идентификатору. Полученные данные

преобразовывались в формат, удобный для отображения в пользовательском интерфейсе, включая перевод категорий, состояний и цветовых характеристик.

## **2.4 Отображение информации об инструменте**

На основе полученных данных была реализована функция отображения детальной информации об инструменте. На странице выводились изображение, название, категория, стоимость аренды, состояние, цвет и ориентация инструмента.

Также был реализован механизм отображения статуса доступности инструмента. В зависимости от полученных данных кнопка бронирования автоматически активировалась или блокировалась, что предотвращало оформление заказа для недоступного оборудования.

## **2.5 Отображение информации о помещении**

Аналогично для помещений была реализована функция отображения детальной информации об помещении. На странице выводились изображение, название, категория, стоимость аренды, особенности и описание.

Отображается и статус помещения. По умолчанию помещение свободно для бронирования, если оно занято на какое-либо время – статус меняется, а кнопка блокируется.

## **2.6 Реализация выбора дат и расчёта стоимости**

Для выбора периода аренды инструментов были реализованы выпадающие списки дат начала и окончания аренды. Списки заполнялись автоматически на основе текущей даты с ограничением максимального периода бронирования. После выбора дат на клиентской стороне выполнялся расчёт количества дней аренды и итоговой стоимости. Расчёт выполнялся динамически без перезагрузки страницы, что позволяло пользователю сразу видеть итоговую сумму аренды.

Для выбора периода аренды студийных помещений были реализованы выпадающие списки даты бронирования, времени начала и количества часов. Последнее зависело от графика работы студии: например, если пользователь просматривает информацию в 18.00 и желает оформить бронирование на этот же день на 19.00, выпадающий список количества часов будет содержать только два варианта: 1 час и 2 часа. На прошедшее время забронировать помещение невозможно. Если пользователь смотрит страницу помещения позже окончания времени работы музыкальной студии, выпадающее меню времени бронирования будет содержать текст «На сегодня времени нет». Итоговая стоимость также рассчитывается динамически.

## **2.7 Реализация механизма бронирования**

После выбора бронируемой позиции и периода аренды была реализована функция оформления бронирования. При нажатии кнопки бронирования данные о заказе формировались на клиентской стороне и временно сохранялись в sessionStorage для передачи между страницами.

Далее пользователь перенаправлялся на страницу подтверждения заказа, где отображалась сводная информация о бронировании, включая название позиции, сроки аренды и итоговую стоимость, включающая в себя основные данные клиента.

## **2.8 Подтверждение бронирования и сохранение данных**

На странице подтверждения заказа была реализована кнопка подтверждения бронирования. При её нажатии выполнялась проверка авторизации пользователя. В случае успешной проверки данные бронирования отправлялись на сервер с использованием HTTP-запроса методом POST.

Серверная часть приложения принимала запрос, выполняла валидацию данных и сохраняла информацию о бронировании в базе данных. После успешного сохранения пользователь получал уведомление о подтверждении заказа и перенаправлялся на страницу с историей бронирований.

## **2.9 Работа с пользовательскими данными и хранилищем браузера**

В ходе разработки активно использовались механизмы локального и сессионного хранилища браузера. В localStorage сохранялись данные авторизованного пользователя и история его бронирований, а в sessionStorage — временные данные текущего заказа.

Использование хранилищ позволило сохранить состояние приложения между страницами без необходимости повторных запросов к серверу, а также упростило передачу данных внутри клиентской части.

## **2.10 Обработка ошибок и тестирование**

На завершающем этапе разработки была реализована обработка возможных ошибок, возникающих при загрузке данных или отправке запросов на сервер. В случае возникновения ошибок пользователю отображались соответствующие уведомления.

Также было проведено ручное тестирование основных сценариев использования приложения, включая просмотр каталога, оформление бронирования и подтверждение заказа. В результате тестирования были выявлены и устранены логические и интерфейсные ошибки.

## **2.11 Вывод по практической части**

В ходе практической части были последовательно реализованы основные функциональные возможности веб-приложения аренды музыкальных инструментов и студийных помещений. Полученные навыки разработки клиентской логики, работы с REST API и обработки пользовательских данных позволили создать работоспособное приложение, соответствующее поставленным требованиям.

## **Заключение**

В ходе прохождения ознакомительной практики была выполнена разработка веб-приложения для аренды музыкальных инструментов и помещений музыкальной студии, направленного на автоматизацию процессов бронирования и учёта оборудования. В рамках поставленных задач был реализован функционал просмотра каталогов, отображения детальной информации, выбора сроков аренды, расчёта стоимости и подтверждения бронирования с сохранением данных в базе данных.

В процессе работы были изучены и применены принципы построения клиент-серверных веб-приложений, организация взаимодействия с REST API и методы обработки пользовательских данных. Особое внимание уделялось реализации клиентской логики, динамической загрузке данных, обработке дат и времени, а также взаимодействию с серверной частью приложения.

Практическая реализация проекта позволила закрепить навыки работы с языком JavaScript, средствами HTML и CSS, а также получить опыт интеграции фронтенд-части с серверной платформой ASP.NET. В ходе выполнения практики были освоены методы отладки, тестирования и обработки ошибок в веб-приложениях.

Разработанное веб-приложение может быть использовано в качестве основы для дальнейшего развития, включая расширение функциональности, улучшение пользовательского интерфейса и внедрение дополнительных возможностей, таких как система уведомлений или онлайн-оплата услуг.

Таким образом, цели и задачи практики были достигнуты, а полученные в ходе работы знания и практические навыки могут быть применены в дальнейшей профессиональной деятельности в области разработки веб-приложений.

## **Список литературы**

1. Документация по продуктам Microsoft: [сайт]. — URL: <https://learn.microsoft.com/ru-ru/> (дата обращения: 08.10.2025).
2. Документация PostgreSQL: [сайт]. — URL: <https://www.postgresql.org/docs/> (дата обращения: 29.09.2025).
3. Metanit.com: руководство по Entity Framework Core: [сайт]. — URL: <https://metanit.com/sharp/efcore/> (дата обращения: 08.10.2025).
4. MDN Web Docs: [сайт]. — URL: <https://developer.mozilla.org/en-US/> (дата обращения: 22.10.2025).
5. Документация Visual Studio Code: [сайт]. — URL: <https://code.visualstudio.com/docs> (дата обращения: 22.10.2025).