МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение высшего образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 05 01-03 Информационные системы и

технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

по дисциплине «Программирование в Internet»

Тема: «WEB-приложение для администрирования студенческого общежития»

Исполнитель

студент 4 курса 1 группы Пыркин С.Ю.

(подпись) (инициалы, фамилия)

Руководитель

Ассистент Харланович А.В.

(подпись) (инициалы, фамилия)

Курсовая работа защищена с оценкой

Минск 2020

**Содержание**

[Введение 4](#_Toc58680922)

[1 Обзор аналогов 6](#_Toc58680923)

[1.1 Preno management system 7](#_Toc58680924)

[1.2 The Booking Factory 10](#_Toc58680925)

[1.3 Sirvoy 12](#_Toc58680926)

[1.4 YuKoSoft Гостиница 14](#_Toc58680927)

[2 Проектирование 17](#_Toc58680928)

[2.1 Функциональные требования 17](#_Toc58680929)

[2.2 Выбор платформы 17](#_Toc58680930)

[2.3 Выбор технологий реализации интерфейса 19](#_Toc58680931)

[2.4 Архитектура проекта 19](#_Toc58680932)

[2.5 Схема базы данных 20](#_Toc58680933)

[3 Разработка приложения 21](#_Toc58680934)

[3.1 Структура проекта 22](#_Toc58680935)

[3.2 Разработка базы данных 24](#_Toc58680936)

[4 Тестирование 25](#_Toc58680937)

[4.1 Регистрация и авторизация 25](#_Toc58680938)

[4.2 Изменение информации о комнате 27](#_Toc58680939)

[5 Руководство по развертыванию приложения 29](#_Toc58680940)

[Заключение 31](#_Toc58680941)

[Список литературы 32](#_Toc58680942)

[ПРИЛОЖЕНИЕ А 33](#_Toc58680943)

[Сервер 33](#_Toc58680944)

[Обработка запросов к студентам 34](#_Toc58680945)

[ПРИЛОЖЕНИЕ Б 37](#_Toc58680946)

[Страница списка комнат 37](#_Toc58680947)

# Введение

Общежитие – место временного проживания для приезжих студентов, учащихся, на время учёбы, сезонных рабочих, на время работы или летней практики. Также существуют рабочие общежитие, но в данной курсовой работе речь пойдет только про студенческие общежития, а конкретнее общежитие университета БГТУ. Как правило, площадь в общежитиях распределяется из норматива 5-6 м. кв. (койкоместо) на одного жильца. В одной комнате могут проживать несколько человек или другое количество, согласно этому нормативу. Уровень комфорта в общежитии обычно довольно низкий. Чаще всего студенческие общежитие разделяют на "мужские" и "женские", но также можно получить комнату в случае, если вместе учится молодая пара.

Общепринято воспринимать студентов как молодых, одиноких, подвижных людей с небольшим достатком. Но есть все возрастающая необходимость иметь дело с более широким кругом людей: студентами с ограниченными физическими возможностями, взрослыми и студентами, имеющими семью и детей, включая семью с одним родителем.

Принципы унифицированного типового проектирования, получившие широкое распространение, сказались на качестве жилой студенческой среды. Виден недостаточный уровень организации жилой студенческой среды по сравнению с зарубежными аналогами. Для современных жилых студенческих комплексов характерен низкий уровень потребительских качеств студенческого жилища.

Студентам в первую очередь должно быть комфортно находиться в комнате, чтобы было приятно заниматься и отдыхать, проводить неформальный академический и социальный обмен с другими студентами, магистрантами и сотрудниками университета, которые проживают в одном общежитии. Также общежитие должно предоставлять студентам возможность уединения и тишины там, где люди живут рядом друг с другом и пользуются общим оборудованием, большинство студентов не имеют право выбирать соседей и даже при заселении на следующий год, не всегда учитываются пожелания в связи с невозможностью организовать весь процесс достаточно качественно и в связи с недостатком жилых мест. В общежитие также заселяют студентов, которые приезжают на соревнования или сокращенные курсы, поэтому свободные комнаты должны быть всегда, а также необходимо контролировать своевременность заселения и выселения студентов, проверять действительно студент нуждается в общежитии или по факту там не проживает. Для многих студентов это первый опыт жизни в дали от дома и необходимость обеспечивать себя самостоятельно, большинству студентов не нравится ощущение жизни в обстановке учебного заведения, но многие ценят ощущение того, что они являются составной частью учебного заведения с его целями и интересами. Студентам приходится самостоятельно платить за жилье, чаще всего его высчитывают из стипендии, в случае, когда студент ее получает, когда же не получает, то необходим способ узнавать, когда и сколько требуется оплатить за жилье, чтобы не платить лишние деньги в случае просрочки или неожиданно узнать, что не заплатил за какой-то месяц проживания или же не доплатил.

Хотелось бы предоставить студентам возможность оперативно уведомлять и оценивать показатели теплоты в помещениях, освещения, наличия горячей воды, чистых общественных помещениях и о состоянии оборудования общего пользования. Также предоставить сотрудникам и администрации возможность вносить в дело студента информацию о проступках и, наоборот, положительных делах, которые влияют на качество и престиж общежития, об общественной пользе, а также личных заслугах.

Чтобы качественно расселить студентов, чтобы не возникало конфликтов интересов и, чтобы студенты учились друг у друга, т. к. это, скорее всего, их первая самостоятельная жизнь вдали от дома и родителей, необходимо иметь характеристику и статистику, которая в этом поможет и позволит улучшить качество жизни студентов на бытовом уровне.

Администрирование позволит быстро менять расположение студентов в общежитии, в случае необходимости найти студента по номеру телефона, фамилии, почте, место расположению, в какой комнате он проживает и с кем, также узнать его возраст, когда он должен выселиться, а когда заселился. Благодаря тому, что все это будет выполняться в одном приложении с одним интерфейсом это позволит сделать унифицированный интерфейс и улучшить опыт пользователей, что сократит время на привыкание к интерфейсу и его возможностям. А также сэкономит много бумаги и времени, а в случае необходимости данные можно экспортировать в документы Excel, а также импортировать в них, распечатать списки, отсортировать или отфильтровать данные, составить отчеты и графики, что повысит эффективность работы и использования данных, а с помощью компьютера не нужно будет вручную писать множество данных, искать студентов по папкам или лично ходить проверять, живет ли там студент, кто его соседи и когда они заселяются, все это можно будет быстро проверить в приложении. Например, можно найти, кто заселяется в определенных датах, а кто выселяется, кому из студентов больше 18 лет, а кому меньше. И множество остальных вариантов использования приложения для администрирования общежития.

Чтобы повысить эффективность и независимость от платформы, а также избавиться от привязки к компьютеру и сделать доступ к данным более мобильным, необходимо разработать веб-приложение, которое позволит осуществлять это. Можно будет непосредственно в самой комнате сверять по спискам проживающих, записывать их на дежурства или передавать любую другую информацию.

Обзор аналогов

В данном разделе я хочу изучить существующие веб и десктоп приложения, которые позволяют управлять общежитием, а также в сравнение будут включены существующие порталы для бронирования комнат, потому как функционал несколько схож с планируемым.

Для того, чтобы рассмотреть аналоги, необходимо продумать, по каким критериям я буду их оценивать. В данном случае исходники приложений недоступны, поэтому для оценки будут рассматриваться только юзабилити характеристики. В качестве критериев будут выступать следующие показатели:

* контент;
* функционал;
* кроссбраузерность или кроссплатформенность;
* навигация;
* дизайн;
* скорость работы.

Теперь поговорим немного про сами характеристики и почему именно они были выбраны в качестве оценочных. Начнем с характеристики под названием «контент», данная характеристика позволяет оценить информацию и тексты, которые размещаются на сайте – основной смысл. Тексты должны быть максимально читабельны и понятны. Если у пользователя возникнут трудности в чтении, понимании, то скорее всего он покинет сайт раньше, чем вы бы этого хотели.

Следующая характеристика «функционал», которая позволяет оценить наличие ошибок, недочётов, а также лишние элементы, которые будут всегда всплывать. Данная характеристика является, пожалуй, одной из самых главных, т. к. она требует, чтобы разработчик своевременной реагировал на критику, замечал и быстрее избавлялся от ненужных функций и добавлял те, что требует пользователь, чтобы не потерять клиента.

За функционалом следует немаловажная характеристика «кроссбраузерность» или «кроссплатформенность», данная характеристика влияет не столько на само содержание приложения, сколько на возможность им воспользоваться потенциальному потребителю. Поэтому приложение должно отображаться и работать во всех популярных браузерах и на всех планируемых платформах одинаково хорошо. Проверить можно самостоятельно, открыв сайт во всех браузерах или установив приложения на целевые платформы и сравнив опыт от взаимодействия.

Характеристикой, которая позволяет оценить, насколько быстро и просто пользователь сможет добираться до конкретных опорных точек функционала и выполнять желаемые действия, является «навигация». Навигация должна быть качественно продумана, чтобы пользователь всегда мог перейти на главную приложения или вернуться на шаг назад, чтобы пользователь имел возможность перейти на смежные страницы, что позволит сэкономить время и повысит эмоциональный настрой от использования.

Одной из характеристик, которая часто является решающей для конечного пользователя, служит «дизайн». Дизайн – это первое, что видит пользователь после открытия приложения, наличие нагромождения, которое повышает время доступа к контенту, чрезмерное количество информации и всплывающих окон, неприятные и непритягательные цвета, а также цвета, которые не соответствуют действию, которое можно было бы ожидать, примером этого являются объемные кнопки, понятные и общепринятые цвета ссылок, а также изменение элемента управления в зависимости от действия, например, при наведении на кнопку меняется стиль курсора, а также сам стиль кнопки, во время нажатия кнопка также должна менять свой стиль, чтобы было понятно, чтобы было произведено действие.

И последняя по очереди, но не по значимости, идет характеристика под названием «скорость работы». Данная характеристика позволяет оценить время, которое затратит пользователь на выполнения тех или иных действий, а также скорость обработки данных. Эта характеристика часто является решающей в случае, когда присутствуют аналоги, возможно уступающие в некоторых других характеристиках, но позволяют оперативно и без лагов выполнить основную работу.

Теперь поговорим непосредственно про

Preno management system

В качестве первого аналога выступает веб приложения для бронирования комнат. Данное приложение предоставляет бесплатную версию на 30 дней, которой я могу воспользоваться для составления оценки.

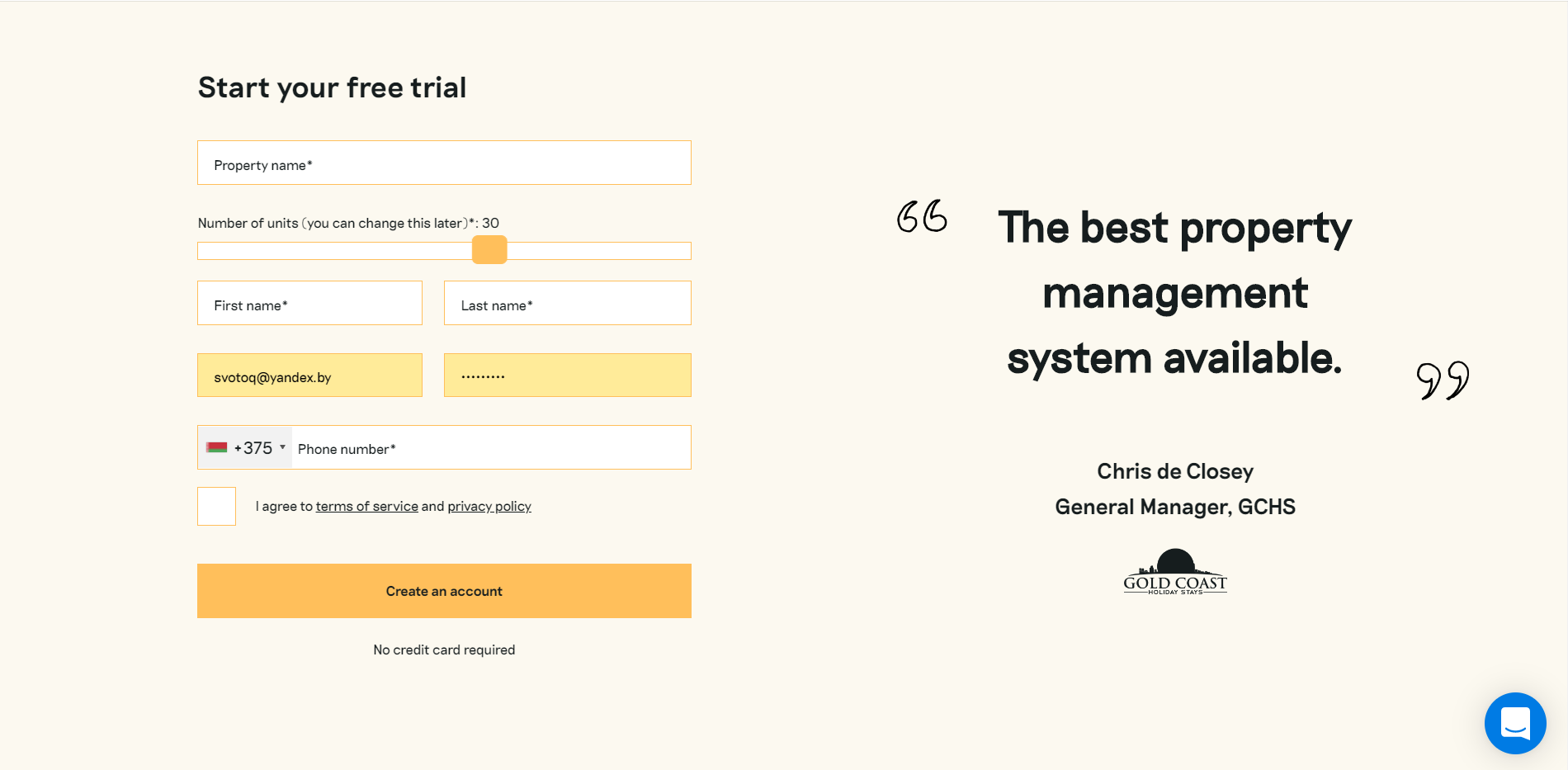


Рисунок 1.1 – Оформление демо версии

После оформления демо версии, приложение перенаправляет на главную страницу приложения без необходимости подтверждать что-либо по почте или номеру телефона.

Следующим предстает главная страница с большой таблицей, с помощью которой и происходит бронирование, то есть это основной функционал, и он вынесен на главную.

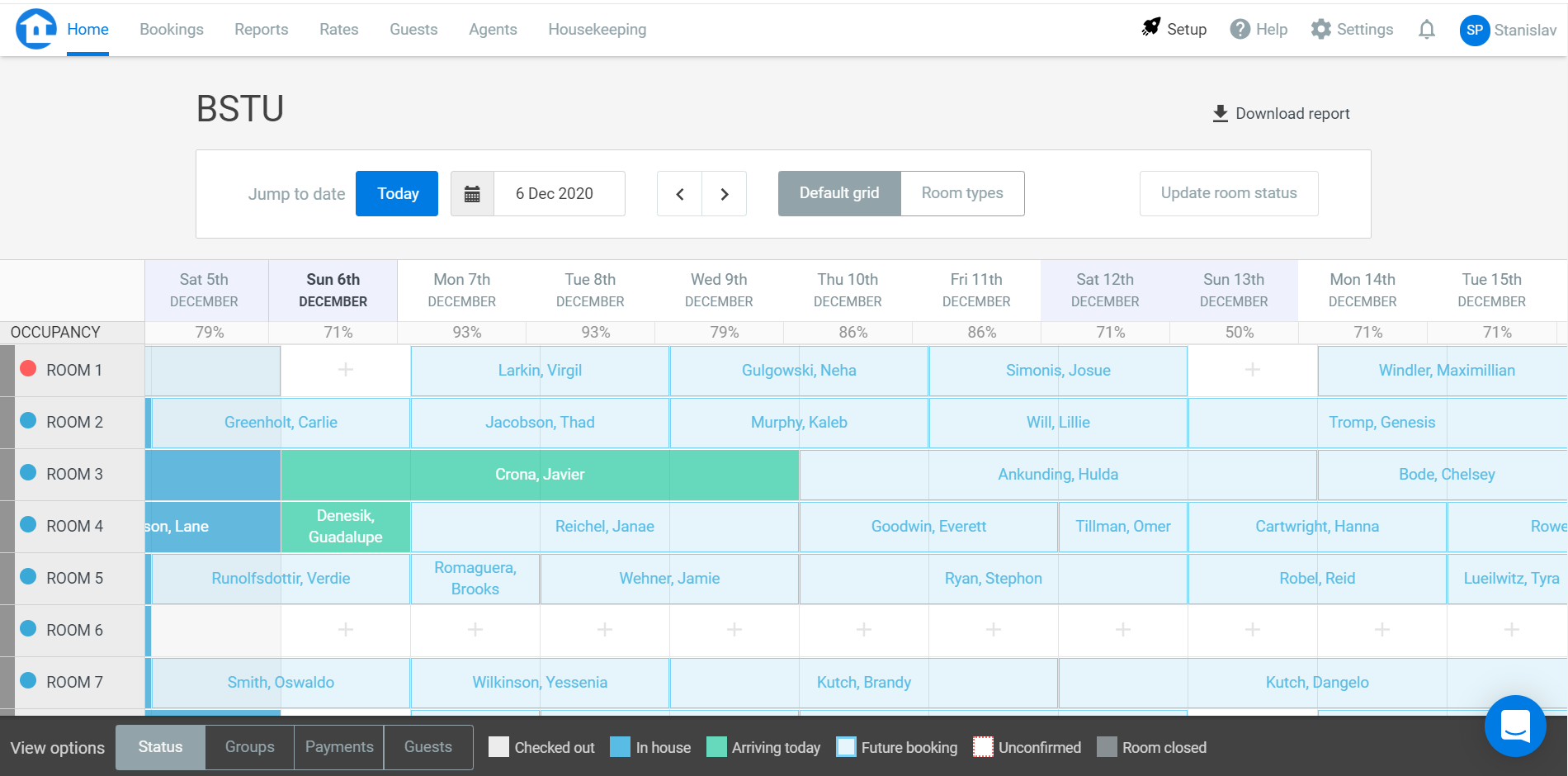


Рисунок 1.2 – Главная страница приложения

При первом входе в приложение легко растеряться из-за большего нагромождения различных цвет и элементов управления, но после нескольких минут тщательного изучения весь функционал становится понятным.

Следует начать с того, что главная страница обладает основным функционалом, простой навигацией по датам и простой системой бронирования, а также доступна фильтрация по типам комнат. Чтобы проще было ориентироваться по таблице, внизу представлена сноска, которая показывает информацию о том, за что каждый цвет отвечает. Несомненным плюсом является кнопка скачивания

Следующим идет меню в самом верху, чтобы можно было переходить на основные вкладки и просматривать информацию о забронированных комнатах, уведомлениях, гостях, которые осуществили бронирование (почту и другие персональные данные), также можно выбрать сервисы, на которых размещены объявления о свободных комнатах, например, «Booking» и «Agoda», для настройки можно указать параметры, такие как такса, которую взымают сервисы за свои услуги и взымается ли эта плата с гостя.

Последней вкладкой является «Housekeeping», которая отвечает за информацию о состоянии жилья, данная вкладка является полезной, т. к. позволяет своевременно отслеживать проблемные комнаты, и кто в них жил, а также своевременно убираться в помещении.

Приложение не заканчивается только на основном функционале, следует рассмотреть наличие документации или возможности обратной связи с разработчиками.

Для этого в приложении была обнаружена вкладка «Help», которая отвечает за возможность задать вопрос или найти ответ среди существующих.

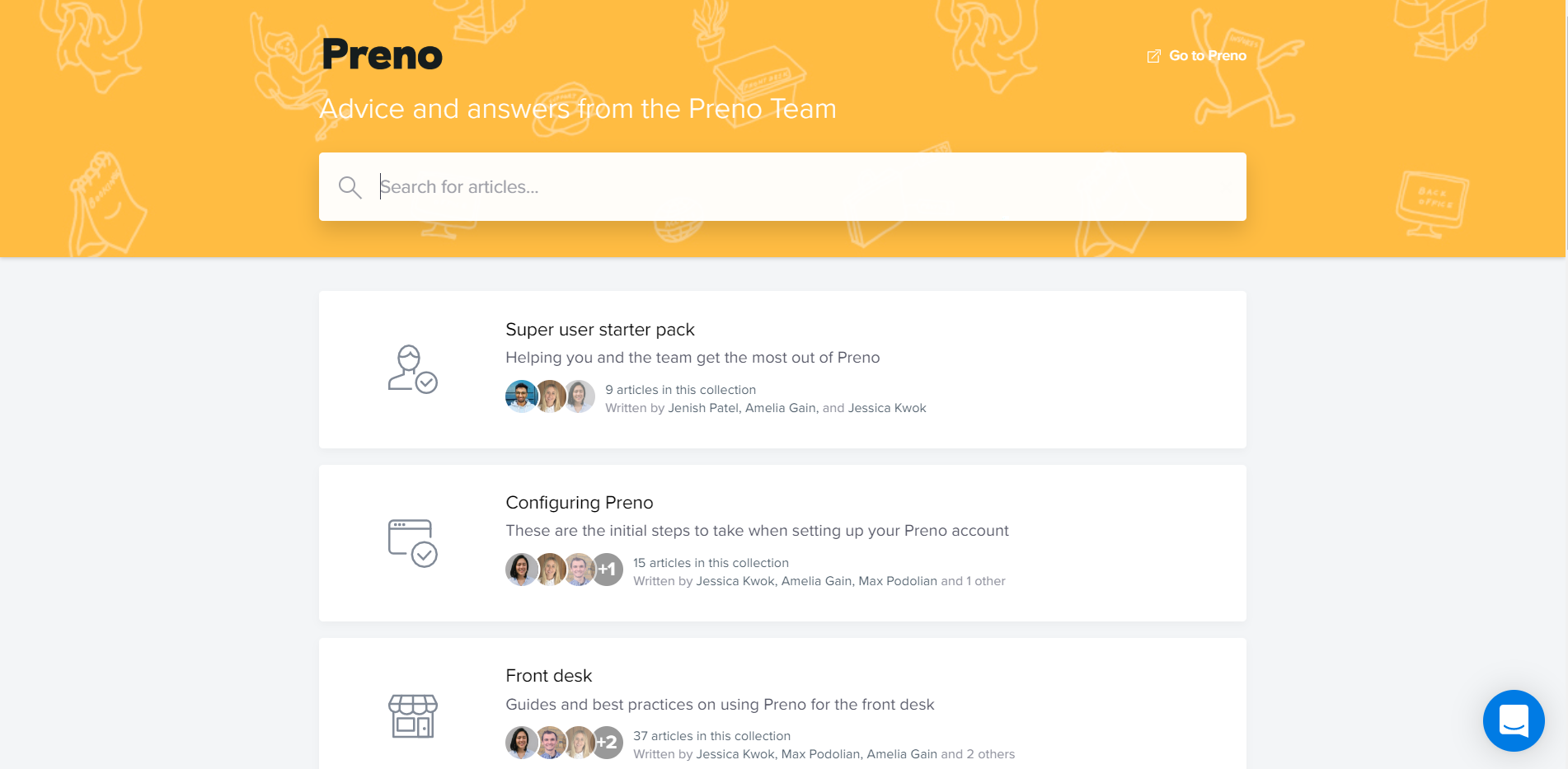


Рисунок 1.3 – Руководство приложения

Несомненным плюсом является то, что помимо документации, приложение предоставляет советы с помощью которых можно повысить эффективность работы и раскрыть все возможности приложения. Отсутствие русского языка или возможности изменить его на любой другой кроме английского это несомненный минус, который отсекает большую часть потенциальных клиентов.

После просмотра всего сайта можно провести оценку и подвести итоги:

* контент на сайте является исключительно рабочим, если не считать главную на которой и происходило оформление демоверсии приложения. Приложение не содержит орфографические или пунктуационные ошибки, название для элементов управления подобраны правильно, снизить бал можно за отсутствие русского языка. Документация является качественной и продуманной, обратная связь возможна с любой из страниц. За контент можно поставить 4 из 5 баллов;
* функционал приложения удовлетворяет всем требованием, наличие графиков и диаграмм, возможность автоматизировать некоторые процессы по отправлению писем, и своеобразная интеграция с популярными сервисами для бронирования позволяет оценить приложение в 5 из 5 баллов.
* кроссбраузерность, приложение работает на всех современных браузерах исключая устаревший Internet Explorer. На мобильных устройствах сайт выглядит неудобным для использования, минимальный размер устройства с которого комфортно пользоваться сайтом это 11 домовый планшет, поэтому сайт заслуживает оценку 4 из 5;
* навигация в приложении достаточно удобная и позволяет с любой вкладки оказаться на любой другой, каждая страничка приложения выполняет только единственную функцию что является несомненным плюсом, поэтому за навигацию приложение заслуживает оценку 5 из 5;
* дизайн в приложении слишком простой, цвета подобраны сильно блеклые и серые, при постоянном использовании можно потерять интерес к жизни, из плюсов стоит отметить шрифт и выдержанную цветовую палитру для всех страниц, в остальном же, дизайн слишком простой и не заслуживает внимания, хотя и не сбивает с толку, поэтому 3 из 5;
* скорость работы, приложение не может удивить моментальной скоростью работы, потому как находится в облаке и некоторые аспекты зависят от скорости интернета, но даже при хорошем интернет-соединении многие функции выполняются долго и могли бы быть более оптимизированными, но при этом не настолько долго, чтобы начать раздражаться, поэтому оценка 3 из 5.

Подводя итоги по данному приложению, он набрал 24 балла из 30 возможных, что является хорошим результатом.

The Booking Factory

Следящее приложение будет также относиться к веб и также предоставляет демоверсию на 30 дней, но в данном случае требуется подтверждение по почте, но после этого предоставляется беспрепятственные доступ к приложению.

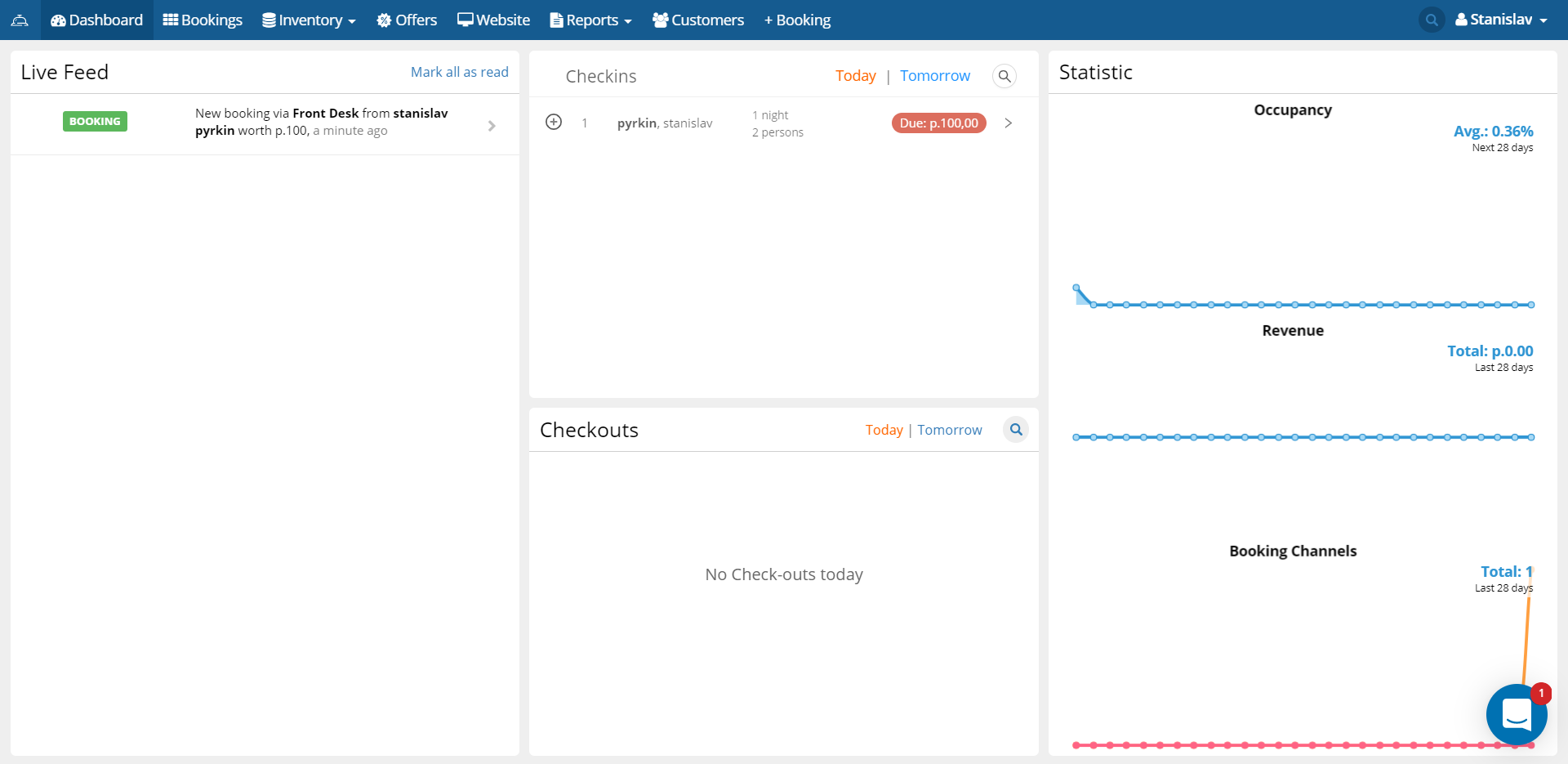


Рисунок 1.4 – Главная страница приложения

Приложение относится к категории бронирования комнат. При входе на главную нас встречает интерфейс, который сразу содержит статистику, новостную ленту, которая отражает последние действия, а также списки забронированных и уже освободившихся комнат.

При первом входе в приложение я слегка растерялся, пришлось потратить некоторое время, чтобы понять функционал, но он достаточно схож с предыдущим приложением. Начнем с оценки контента, в общем, он достаточно отражает и вполне советует ожиданиям, но есть некоторые моменты, вроде «Inventory» или другие неочевидные названия, также отсутствует подсвечивание обязательных полей, поэтому, чтобы забронировать комнату или создать акцию, ушло много времени, чтобы понять почему кнопка «Save» недоступна, документация точно такая же как и в предыдущем приложении, доступен только английский язык, поэтому я оцениваю в 3 из 5 баллов.

Функционал приложения очень порадовал, здесь присутствуют дополнительные функции, которых не было в предыдущем приложении, вроде создания акций, добавления в заказ обеда и т. п. Что очень удивило, так это наличие конструктора для создания своего собственного сайта.

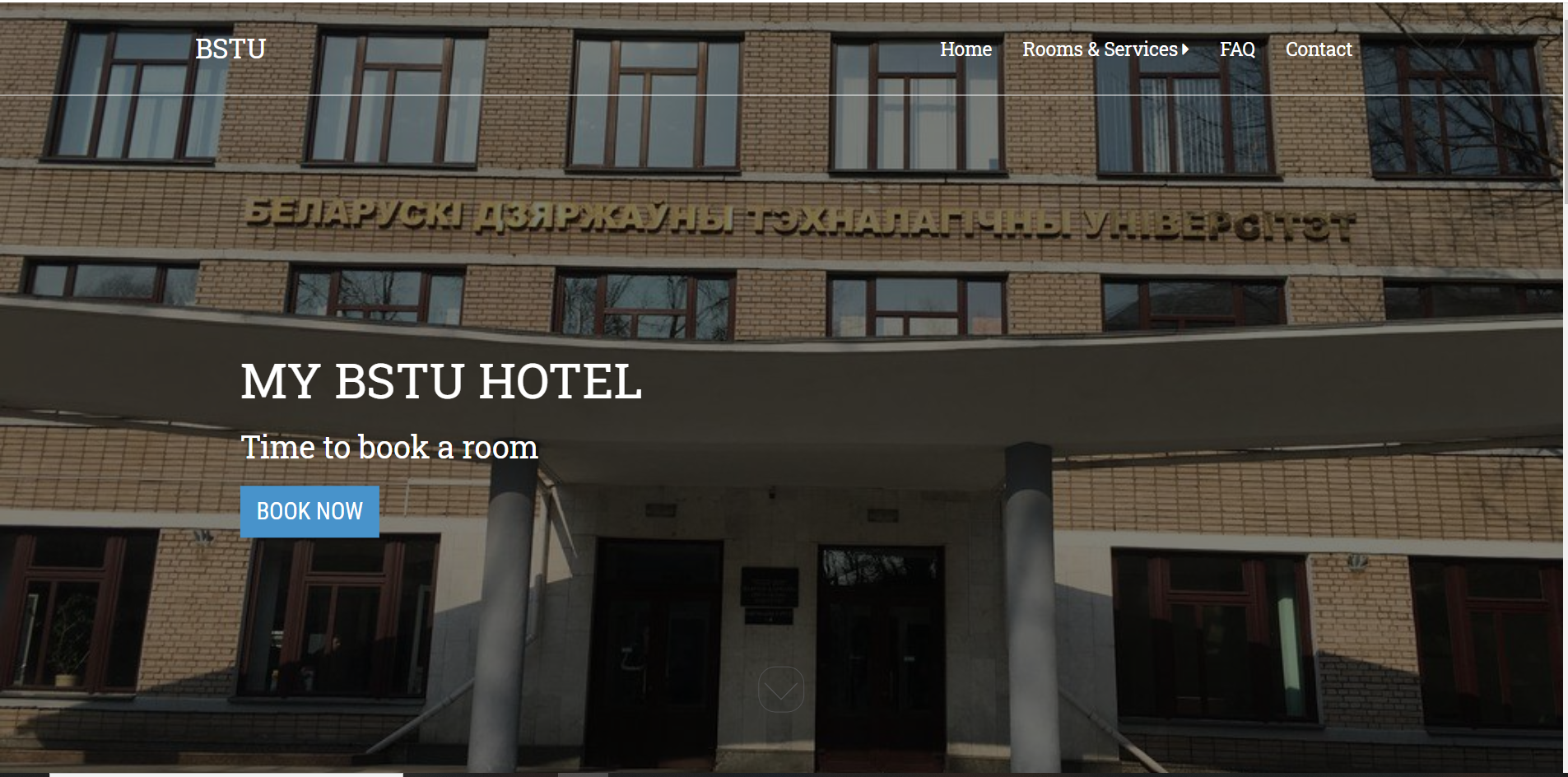


Рисунок 1.5 – Собственный сайт

Функционал создания сайта очень гибкий и позволяет использовать не только встроенную систему бронирования, но и собственную. При этом для создания сайта присутствует более 5 шаблонов с помощью которых можно изменить внешний вид сайта. Создание сайта очень простое, необходимо ввести определенный текст в те места, где он должен быть и загрузить свои картинки. Функционал выше всяких похвал, однозначно 5 из 5 баллов.

Настала очередь кроссбраузерности и кроссплатформенности, основное приложения для управления бронированием не оптимизировано под мобильные устройства, а сайт, который можно создать из шаблона вполне удобен при использовании телефона. Сайт одинаково выглядит во всех браузерах, но отсутствие удобного интерфейса при использовании телефона или планшета отнимает два балла, поэтому 3 из 5 возможных баллов.

Навигация в приложении качественная и позволяет вернуться или перейти на нужную вкладку достаточно быстро и понятно, весь функционал доступен с главной панели, которая видна на всех страницах, навигация заслуживает оценки 5 из 5 баллов.

Дизайн в приложении более интересный, нежели в предыдущем, но оно все также использует не очень яркие и насыщенные цвета, а также их сочетания, что свидетельствует о незамысловатом дизайне, а также некоторые элементы выполнены недостаточно корректно, что затрудняет использование, поэтому 2 из 5 баллов.

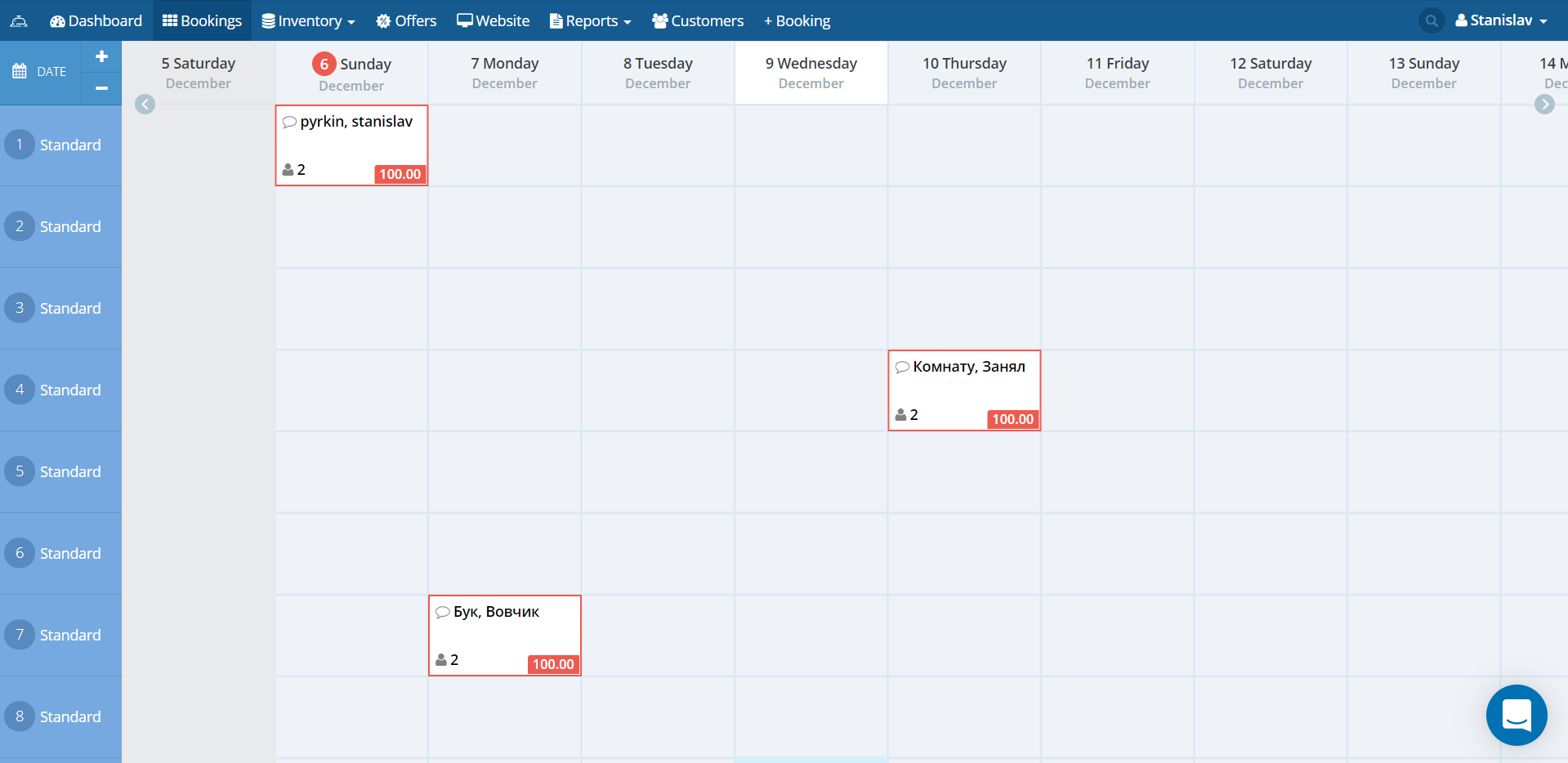


Рисунок 1.6 – Страница бронирования

Страница бронирования визуально отличается и находится на отдельной странице в отличии от предыдущего приложения, что является несомненным плюсом, а также занимает всю доступную площадь и таблица более удобная для использования.

Скорость работы приложения и собственного сайта, который можно создать с помощью конструктора достаточная высокая, но при этом некоторые неочевидные вещи подтормаживают, поэтому можно оценить в 4.5 из 5 баллов.

Итоговая оценка приложения составила 22,5 балла из 30 возможных, что немного хуже предыдущего приложения.

Sirvoy

Следующее приложение предоставляет более обширные возможности и настройки по бронированию. Данное приложение предоставляет тестовый период на 14 дней, что значительно меньше, чем у его конкурентов. Для ознакомления с возможностями этого времени вполне достаточно, но недостаточно будет проверить его на практике и целесообразность оформить подписку.

При входе в приложение нас встречает главная страничка, на которой можно сразу увидеть статистику и выполнить поиск по параметрам.

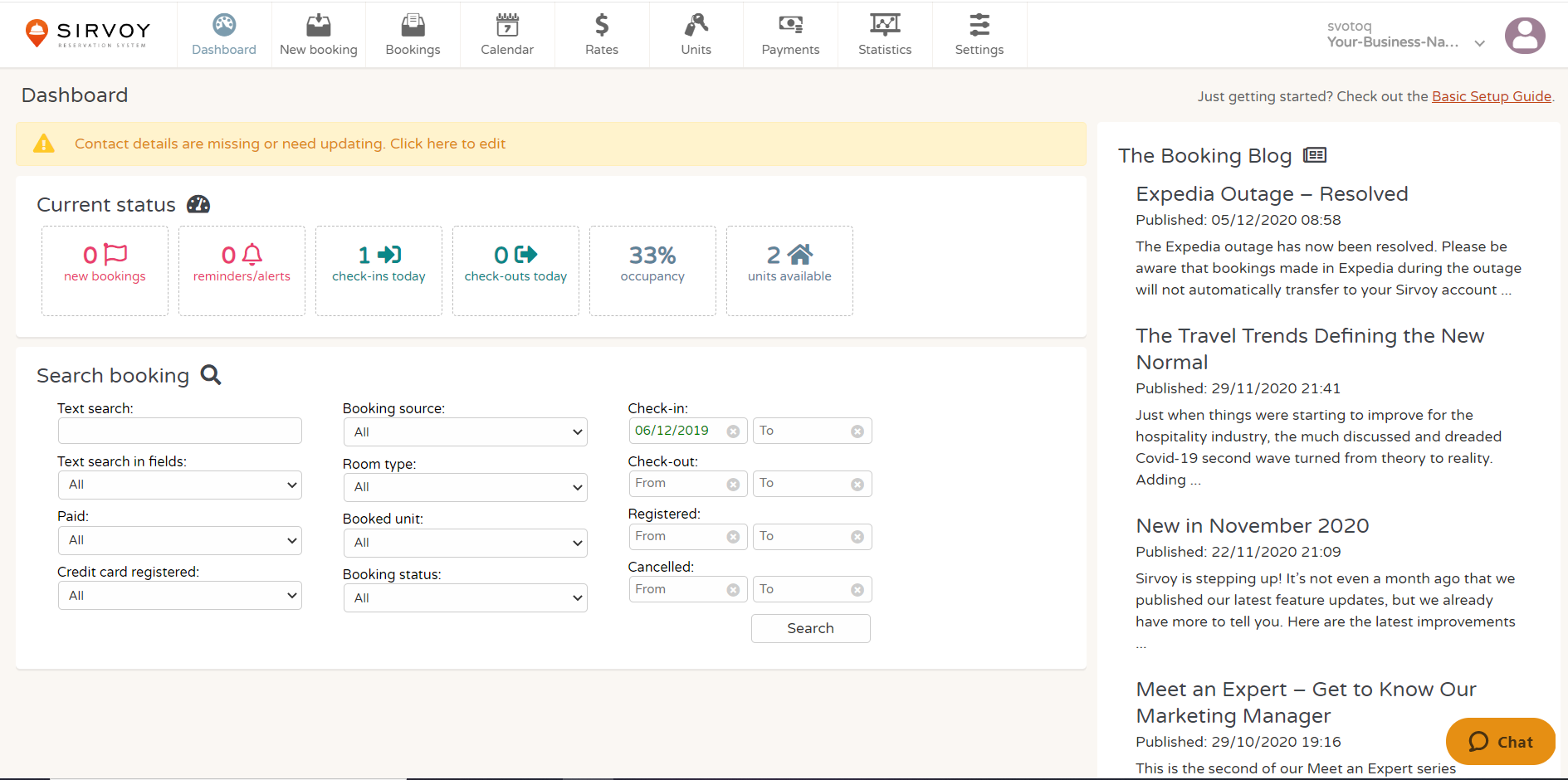


Рисунок 1.7 – Главная приложения

Сразу перейдем к оцениванию приложения, начнем с контента. В приложении очень много текста и параметров, а также присутствует новостная лента, которую нельзя отключить, что является не очень удобным и полезным для некоторых пользователей. Следующее на что хотелось бы обратить внимание это инструкция, она очень краткая и не очень понятная, поэтому у меня возникли большие сложности во время первой настройки и выполнения бронирования, за контент однозначно ставлю 2 из 5 баллов.

Функционал в приложение как это не странно достаточно обширный со множеством критериев и параметров, возможно многофункциональность – это хорошее решение, но без возможности отключать ненужные функции, чтобы сосредоточится только на необходимых является минусом. Основные функции реализованы в полной мере, поэтому 4 из 5 баллов.

Кроссбраузерность в приложении реализована в полной мере. Кроссплатформенность реализована не в полной мере, приложение работает корректно на мобильных устройствах, но из-за большего числа полей и нагромождения, пользоваться на мобильном телефоне через чур неудобно, а вот с планшетом приложение совместимо в полной мере, поэтому итоговая оценка 4 из 5 баллов.

Навигация в приложении не совсем интуитивно понятная и не всегда попадаешь туда, куда нужно, а также не все действия получается выполнить сразу т. к. нет последовательности для выполнения действий, которые оказываются обязательными, за навигацию ставлю 3 из 5 баллов.

Дизайн выполнен в серо-белых тонах, что создает дискомфорт при использовании, зато уведомления на таком фоне хорошо заметны. Стоит отметить наличие валидации полей и их подсветку, что позволяет очень быстро сориентироваться. В остальном дизайн на твердую 3 из 5 баллов, пользоваться можно, но все очень тускло.

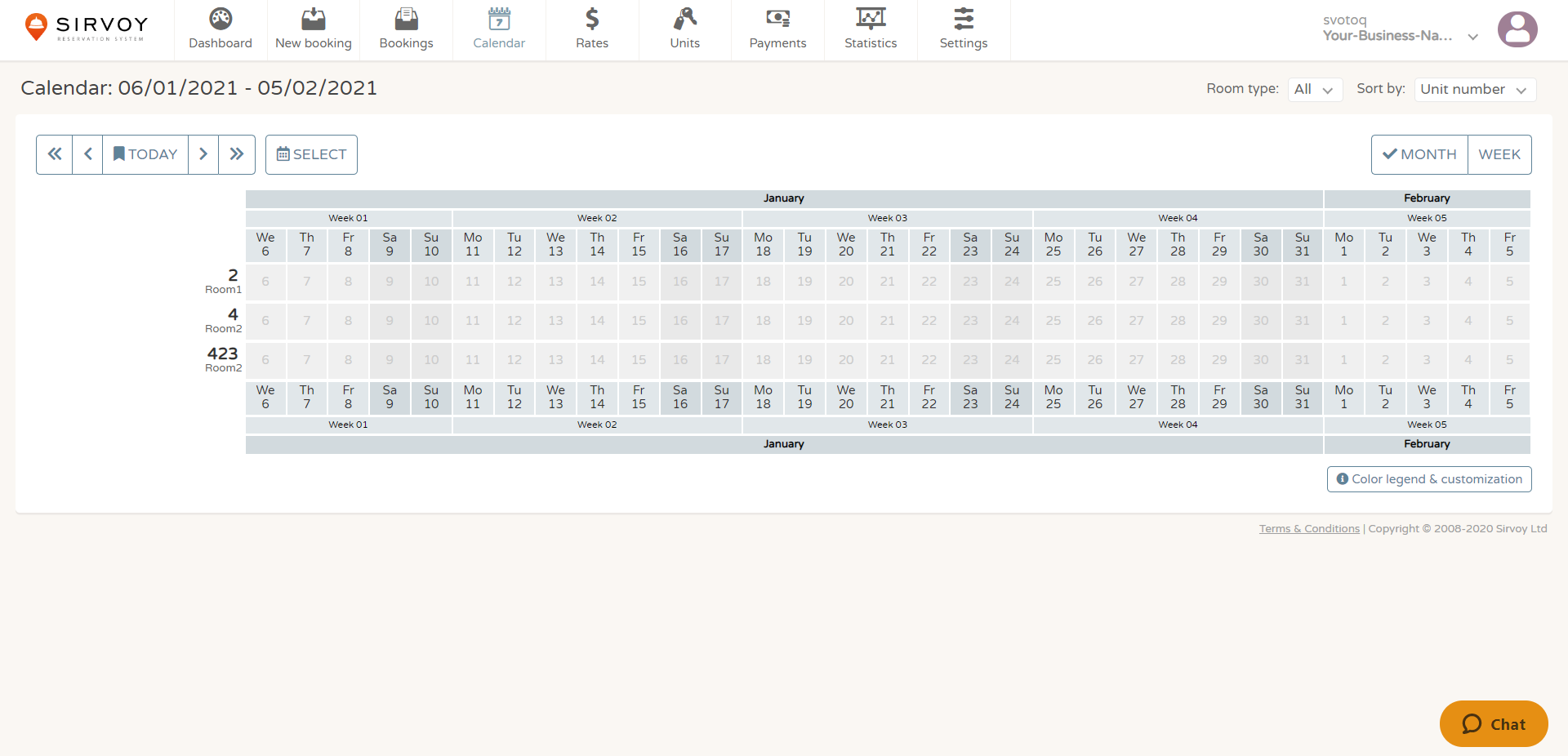


Рисунок 1.8 – Календарь бронирования

Скорость работы приложения приемлемая и ничего не подвисает на незначительных действиях, скорость поиска достаточно высокая, возможно при наличии большой базы данных скорость будет ниже, но в сравнении с конкурентами, данное приложение лидирует по скорости, поэтому твердые 5 баллов.

Подводя итоги по данному приложению, хочется отметить, что оно отличается от предыдущих визуально и имеет значительно больший функционал, хотя и слегка запутанный, конечная оценка 21 балл из 30 возможных.

YuKoSoft Гостиница

Последним идет приложение для персональных компьютеров. Оно предоставляет бесплатную полную версию до 60 запусков приложения. После 60 запусков действуют некоторые ограничения.

Главное окно состоит из трех областей – это навигация, вкладки, которые были выбраны через навигацию, и окно заметок. Удобным является то, что можно открыть несколько окон и быстро переключатся между ними, что позволит оставить только необходимый функционал, возможность сворачивать подпункты меню – это также хорошее решение.

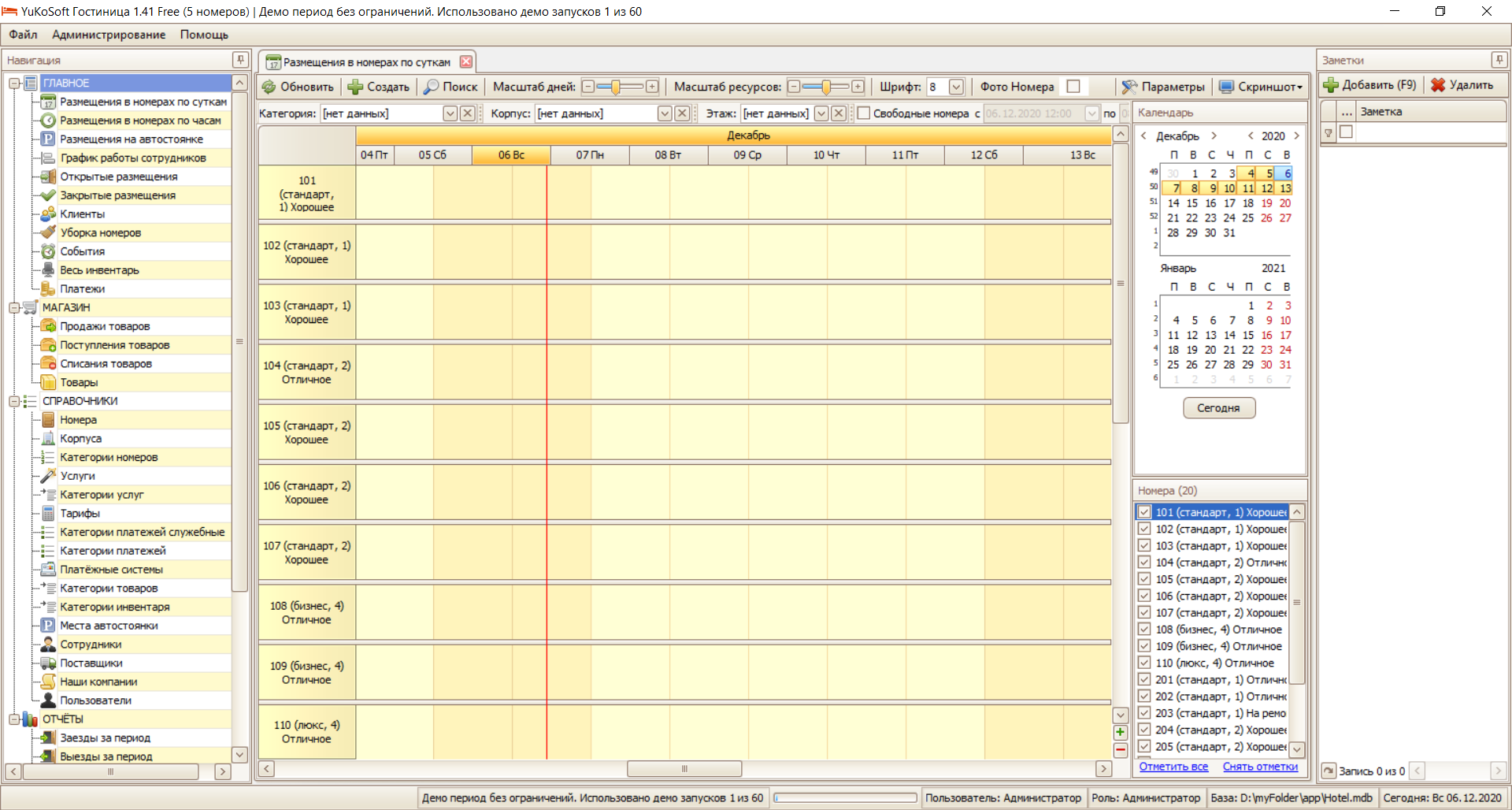


Рисунок 1.9 – Главная приложения

Контент приложения полностью на русском языке без возможности изменения его. Все пункты приложения описаны верно, документации для приложения не предусмотрено, но интерфейс достаточно прост и понятен, поэтому оценка составляет 4 из 5 баллов.

Функционал приложения позволяет сделать все необходимые для администрирования действия, возможность настраивать отображение необходимого функционала и наличие вкладок, возможность экспортировать и импортировать данные и графики, отслеживание графиков сотрудников, учет времени прибытия, использования автостоянки, оценка 5 из 5 баллов.

Кроссплатформенность у данного приложения плохая, т. к. оно доступно только на компьютерах, которые работают на операционной системе Windows. Оценка приложению 2 из 5 баллов.

Навигация в приложении очень простая и доступная со всех окон, возможность сворачивать ненужные пункты, вкладки отображают ровно то, что ожидается от приложения, поэтому за навигацию ставлю 5 баллов.

Дизайн у приложения очень старый и простой, цвета подобраны неудачно, многие иконки слишком мелкие, а другие слишком большие, некоторые элементы интерфейса отображаются непонятно для пользователя, но пользоваться приложением, в целом, удобно и приятно, поэтому оценка 3 из 5 баллов.

Приложение работает с задержками и лагами, некоторые действия выполняются непозволительно долго, открытие вкладок происходит резко, в ходе использования приложение несколько раз зависало, не уведомляя пользователя, что происходят какие-то операции, оценка 2 из 5 баллов.

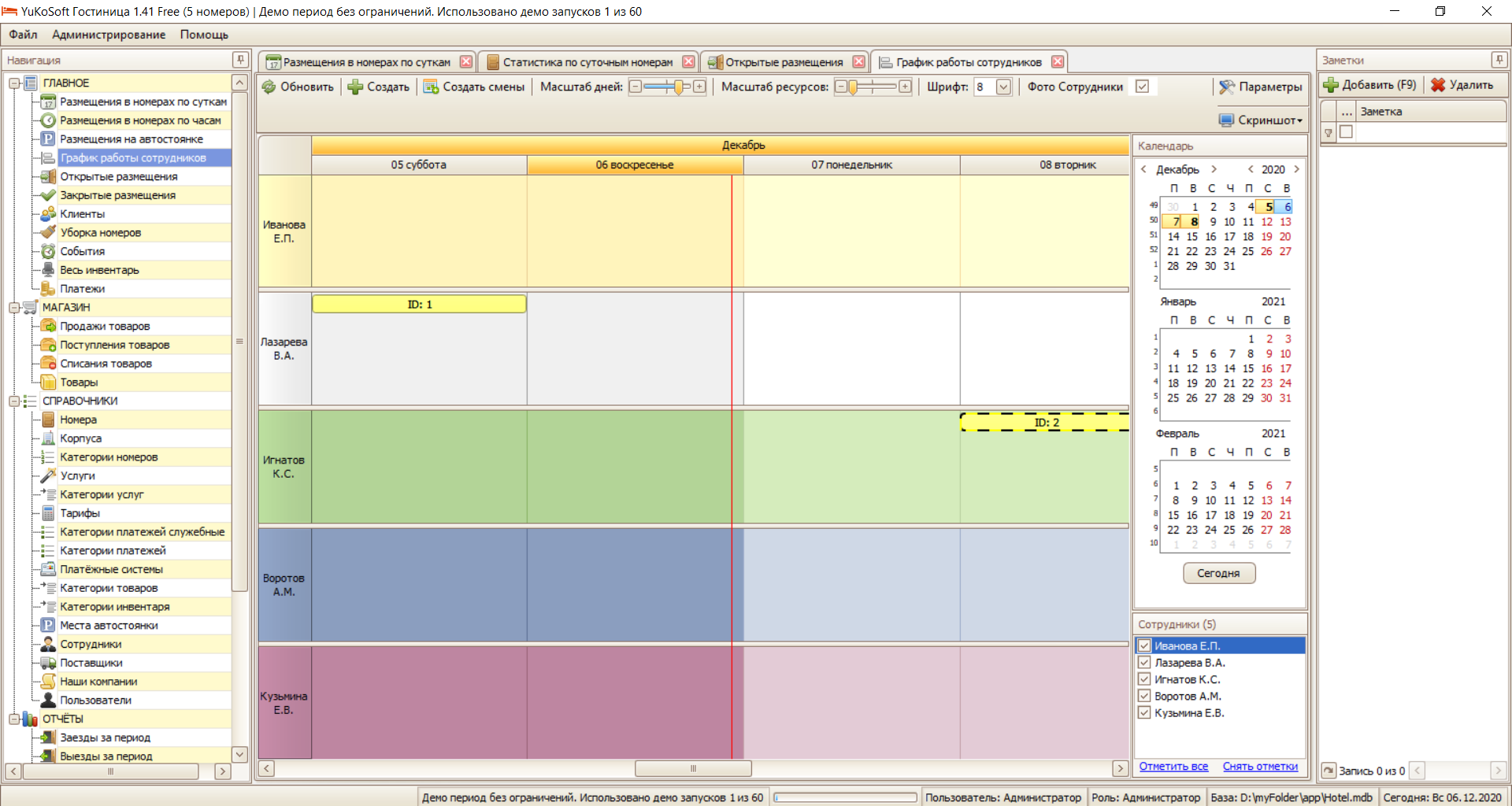


Рисунок 1.10 – График работы сотрудников

Суммирование оценок всех характеристик приложения приносят ему 21 балл из 30 возможных.

После оценки всех приложений можно сделать вывод, что многие имеют схожий интерфейс и концепцию, но каждое приложение реализовало его по-своему. В ходе изучения аналогов, больше всего не понравились приложения, которые были не дружелюбны с пользователем, то есть не содержали подсветки полей, которые являются обязательными, не уведомляли пользователей о наличии ошибок. Приложения, которые обладали излишним функционалом, но при этом не позволяли настраивать видимость также вызывали негативную реакцию. В разрабатываемом приложении следует обратить внимание на то, чтобы оно работало ожидаемо быстро в тех действиях, которые не должны вызывать задержек, не зависало без уведомления пользователя, что идет загрузка чего-либо и было дружелюбным. Для оформления приложения стоит подобрать яркую и в то же время мягкую цветовую палитру, с которой можно работать несколько часов и не уставать. Следует реализовать возможность поэтапного выполнения действий, что позволяло фокусироваться на единственной задачи, избегать расфокуса внимания и было дружелюбно для нового пользователя, который еще не был знаком с данной или похожими системами ранее, но стоит учесть общий концепт приложений, чтобы пользовали, которые имеют опыт использования могли быстрее адаптироваться.

Проектирование

В этом разделе будет рассматриваться информация, связанная с проектированием приложения. Для начала разберем функциональные требования, то есть что приложение должно выполнять, следующим этапом будет выбор платформы, на которой будет строиться приложение, технологии, которые будут использоваться в приложении, а также диаграммы использования и схема базы данных

Функциональные требования

Перед созданием приложения я выделил следующие требования, которые позволят охватить весь перечень задач и обеспечить комфортную работу:

* создание комнат двух типов, складские и жилые помещения;
* заселение студентов в комнаты;
* экспорт данных в excel;
* импорт данных из excel;
* просмотр информации о комнате и студенте;
* редактирование информации о комнате и студенте;
* поиск по комнатам и студентам;
* регистрация и авторизация администраторов;

Выделены были именно данные требования, т. к. они позволят максимально охватить область применения приложения и позволят создавать большую базу данных.

Также следует выдели саму концепцию приложения:

* создание централизированной базы данных;
* разработка серверной и пользовательской частей веб-приложения;
* предоставление API приложения;
* настройка параметров маршрутизации страниц;
* разработка алгоритма авторизации и регистрации администраторов;
* создание простого и понятного интерфейса;

Теперь можно определиться с архитектурой, инструментами разработки и приступать к реализации проекта.

Выбор платформы

В качестве платформы разработки для серверной части была выбрана платформа Node.js. программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Одной из основных привлекательных особенностей Node.js является скорость. JavaScript-код, выполняемый в среде Node.js, может быть в два раза быстрее, чем код, написанный на компилируемых языках, вроде C или Java, и на порядки быстрее интерпретируемых языков наподобие Python или Ruby. Причиной подобного является неблокирующая архитектура платформы, а конкретные результаты зависят от используемых тестов производительности, но, в целом, Node.js — это очень быстрая платформа.

В качестве основного языка разработки был выбран JavaScript, потому что в среде Node.js выполняется код, написанный на JavaScript. Это означает, что миллионы разработчиков пользовательского интерфейса, которые уже пользуются JavaScript в браузере, могут писать и серверный, и клиентский код на одном и том же языке программирования без необходимости изучать совершенно новый инструмент для перехода к серверной разработке, что делает этот язык универсальным. В браузере и на сервере используются одинаковые концепции языка. Кроме того, в Node.js можно оперативно переходить на использование новых стандартов ECMAScript по мере их реализации на платформе. Для этого не нужно ждать до тех пор, пока пользователи обновят браузеры, так как Node.js — это серверная среда, которую полностью контролирует разработчик. В результате новые возможности языка оказываются доступными при установке поддерживающей их версии Node.js. Также, стоит отметить, что немаловажным фактором выбора платформы Node.js является асинхронность выполнения процедур.

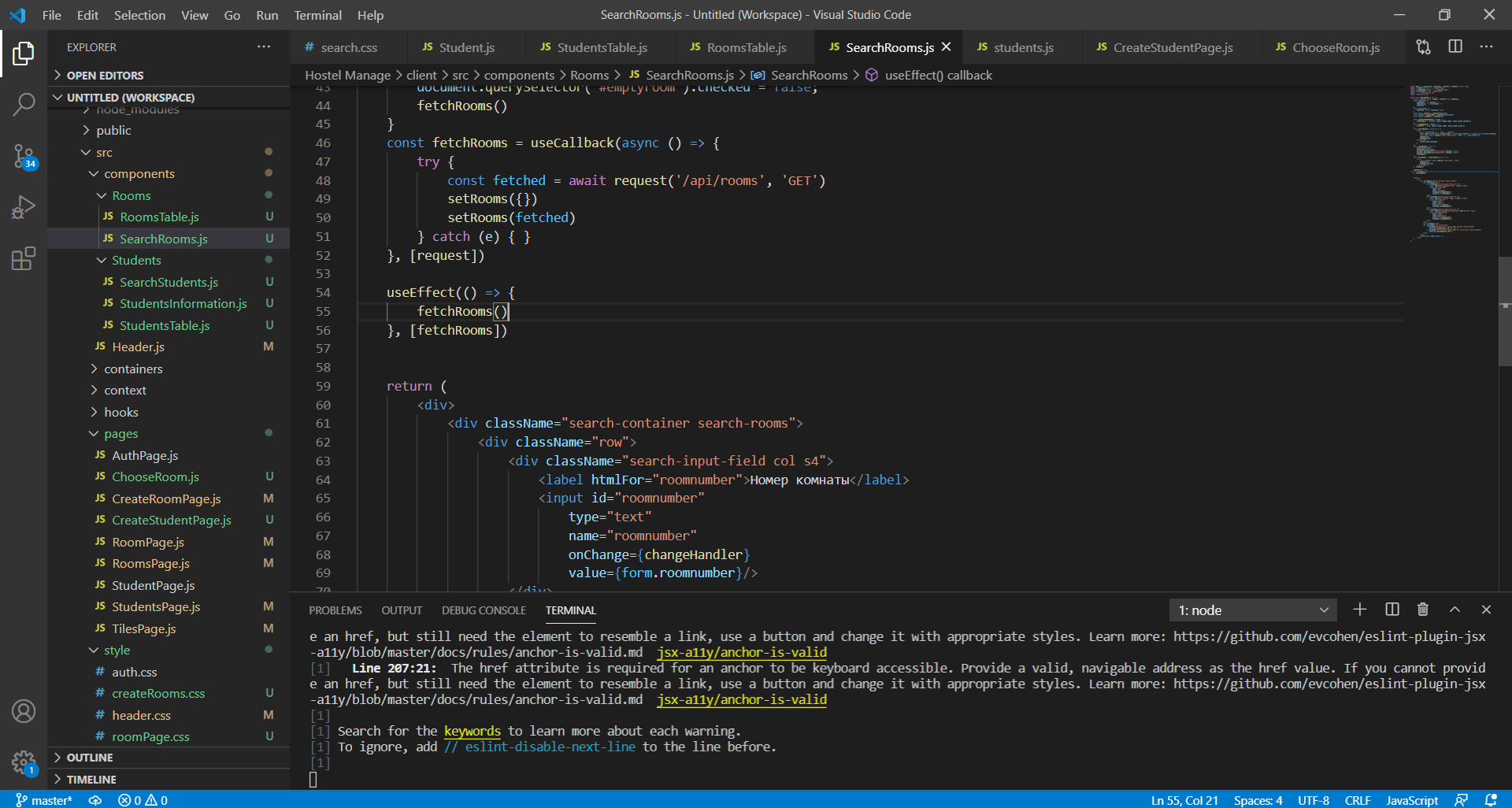


Рисунок 2.1 – Среда разработки

В качестве среды разработки был выбран Visual Studio Code, преимуществом является то, что эта среда разработки может быть быстро установлена, поддерживает множество плагинов, контроль версий, а также позволяет разделять рабочие области и вызывать консоль для запуска и работы с npm пакетами.

Выбор технологий реализации интерфейса

В качестве технологий для реализации пользовательского интерфейса была выбрана библиотека React. Эта библиотека позволяет создавать качественный интерфейс с жизненным циклом объектов, выполнять асинхронный код и использовать функции обратного вызова, создавать компоненты, которые можно будет повторно использовать, встраиваться JavaScript код для генерации элементов управления благодаря страничкам JSX - то расширение синтаксиса JavaScript, которое позволяет использовать HTML-подобный синтаксис для описания структуры интерфейса. Также наличие хуков, которые позволяют использовать состояние и другие возможности React без написания классов. Построение пользовательских хуков позволяет помещать логику компонента в повторно используемые функции. React использует виртуальный DOM. Он создаёт кэш-структуру в памяти, что позволяет вычислять разницу между предыдущим и текущим состояниями интерфейса для оптимального обновления DOM браузера. Таким образом программист может работать со страницей, считая, что она обновляется вся, но библиотека самостоятельно решает, какие компоненты страницы необходимо обновить.

Архитектура проекта

Для разработки проекта будет использоваться REST подход, который на сегодняшний день практически вытеснил все остальные подходы, в том числе дизайн основанный на SOAP и WSDL

REST подход дает следующие преимущества:

* масштабируемость взаимодействия компонентов системы (приложения);
* общность интерфейсов;
* независимое внедрение компонентов;
* промежуточные компоненты, снижающие задержку, усиливающие безопасностью.

Использовать REST следует в следующих ситуациях:

* когда есть ограничение пропускной способности соединения
* если необходимо кэшировать запросы;
* если система предполагает значительное масштабирование;
* в сервисах, использующих AJAX.

REST предоставляет отсутствие дополнительных внутренних прослоек, что означает передачу данных в том же виде, что и сами данные, т. е. данные не оборачиваются в XML, как это делает SOAP и XML-RPC, не используется AMF, как это делает Flash и т. д. Просто отдаются сами данные.

Каждая единица информации (ресурс) однозначно определяется URL – это значит, что URL по сути является первичным ключом для единицы данных. Причем совершенно не имеет значения, в каком формате находятся данные по адресу — это может быть и HTML, и jpeg, и документ Microsoft Word.

Как происходит управление информацией ресурса – это целиком и полностью основывается на протоколе передачи данных. Наиболее распространенный протокол конечно же HTTP. Для HTTP действие над данными задается с помощью методов:

* GET (получить);
* PUT (добавить, заменить);
* POST (добавить, изменить, удалить);
* DELETE (удалить).

Таким образом, действия CRUD (Create-Read-Update-Delete) могут выполняться как со всеми 4-мя методами, так и только с помощью GET и POST.

Чтобы распределенная система считалась сконструированной по REST архитектуре (Restful), необходимо, чтобы она удовлетворяла следующим критериям:

* Client-Server. Система должна быть разделена на клиентов и на серверов. Разделение интерфейсов означает, что, например, клиенты не связаны с хранением данных, которое остается внутри каждого сервера, так что мобильность кода клиента улучшается. Серверы не связаны с интерфейсом пользователя или состоянием, так что серверы могут быть проще и масштабируемы. Серверы и клиенты могут быть заменяемы и разрабатываться независимо, пока интерфейс не изменяется.
* Stateless. Сервер не должен хранить какой-либо информации о клиентах. В запросе должна храниться вся необходимая информация для обработки запроса и если необходимо, идентификации клиента.
* Cache․ Каждый ответ должен быть отмечен является ли он кэшируемым или нет, для предотвращения повторного использования клиентами устаревших или некорректных данных в ответ на дальнейшие запросы.
* Uniform Interface. Единый интерфейс определяет интерфейс между клиентами и серверами. Это упрощает и отделяет архитектуру, которая позволяет каждой части развиваться самостоятельно.

Схема базы данных

При разработке приложений большую роль играет база данных. Чтобы она была правильно составлена, требуется проводить грамотный анализ предметной области приложения.

Результатом корректно проведённого анализа, проектирования, и разработки, является модель, способная предоставить функционально все необходимые возможности для пользователя. В моем случае были выделены следующие основные требования к таблицам и функционалу, который они смогут покрыть, ниже представлен список таких требований:

* возможность регистрации и авторизации администраторов;
* возможность создания студентов;
* возможность создания комнат;
* возможность привязки студента к конкретной комнате.

Основой инфраструктуры базы данных является грамотно спроектированная модель, которая отображает связь пользовательских таблиц. Правильное и корректное взаимодействие их друг с другом как раз и заключается в схеме базы данных со связями, верно отображающими их положение.

На рисунке 2.2 изображена схема базы данных, разработанная при подготовке к созданию веб-приложения и реализацией необходимого функционала. Между сущностями, представленными таблицами, также настроены связи, отражающие их взаимосвязь.

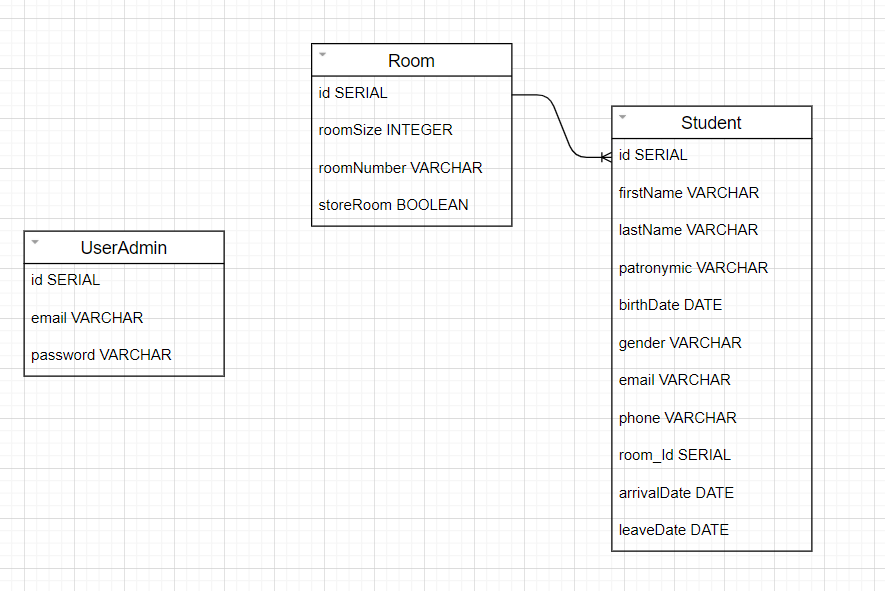


Рисунок 2.2 – Схема базы данных

На схеме отражена одна единственная связь между таблицами – это один ко многим между комнатой и студентами. Данная связь отражает возможность заселения студентов в конкретную комнату. Таблица UserAdmin не содержит связей, т. к. для всех пользователей будет использоваться одна база данных, а вот доступ к этим данным будет предоставляться только авторизированным пользователям, для этого и необходим данная таблица.

# Разработка приложения

В этом разделе будут описываться основные этапы разработки и пояснения к ним, за что отвечает каждый компонент, а также приводиться скриншоты в пояснение и для визуальной демонстрации.

Структура проекта

Серверная часть представляет собой основной файл package.json, которые содержит все зависимости и команды для запуска сервера и пользовательского интерфейса. В файле server.js располагается код для создания сервера и регистрации маршрутов. Следующим идет папка с маршрутами routes, которая содержит в себе API для взаимодействия с данными приложения.

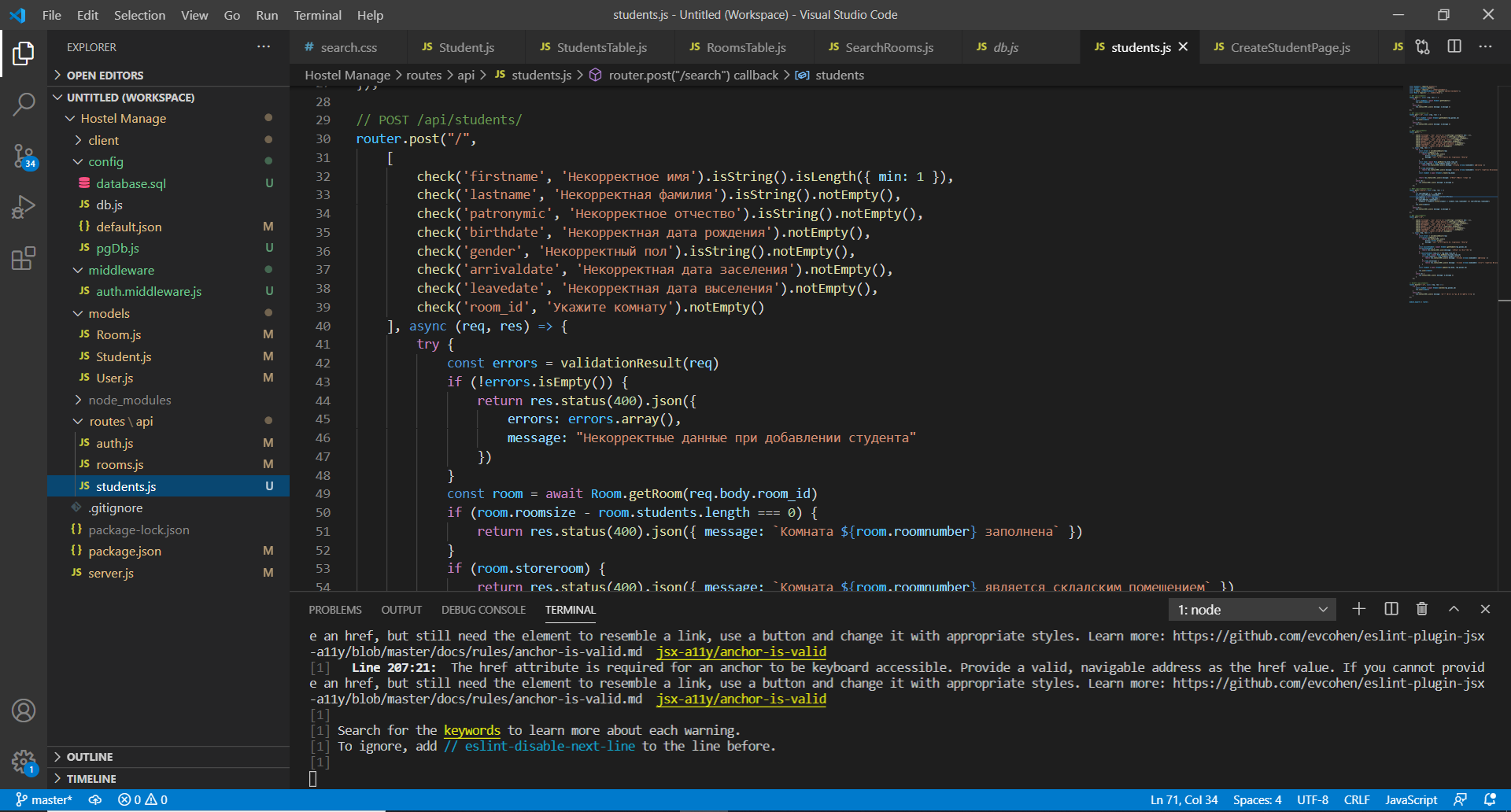


Рисунок 3.1 – Структура сервера

Папка models содержит в себе модели, которые существуют в приложении и способ их взаимодействия с базой данных. Папка middleware отвечает за то, чтобы только авторизированный пользователь мог использовать основные функции приложения. Папка config содержит в себе глобальные параметры приложения, данные для авторизации в базе данных и скрипты для создания базы данных.

Следующим идет архитектура пользовательского интерфейса, которая содержит в себе публичные компоненты вроде логотипа. В папке pages располагаются основные страницы, которые используются при навигации, в файле routes располагаются маршруты для авторизированных пользователей и для тех, кто не авторизирован, для пользователей, которые не авторизированы доступны только страницы входа и регистрации. В папке контекст располагаются данные вошедшего пользователя. В папке hooks располагаются вспомогательные элементы для авторизации, запросов на сервер и вывода всплывающий уведомлений. В папке контейнеры находится один основной элемент, который отвечает за корень приложения. В папке компоненты находятся элементы страницы, которые можно повторно использовать. Стили приложения заданы в папке styles. Также для приложения существует свой собственный package.json в котором находятся зависимости конкретно для клиентской части. Файлы index.js и index.html это точки входа в приложение.

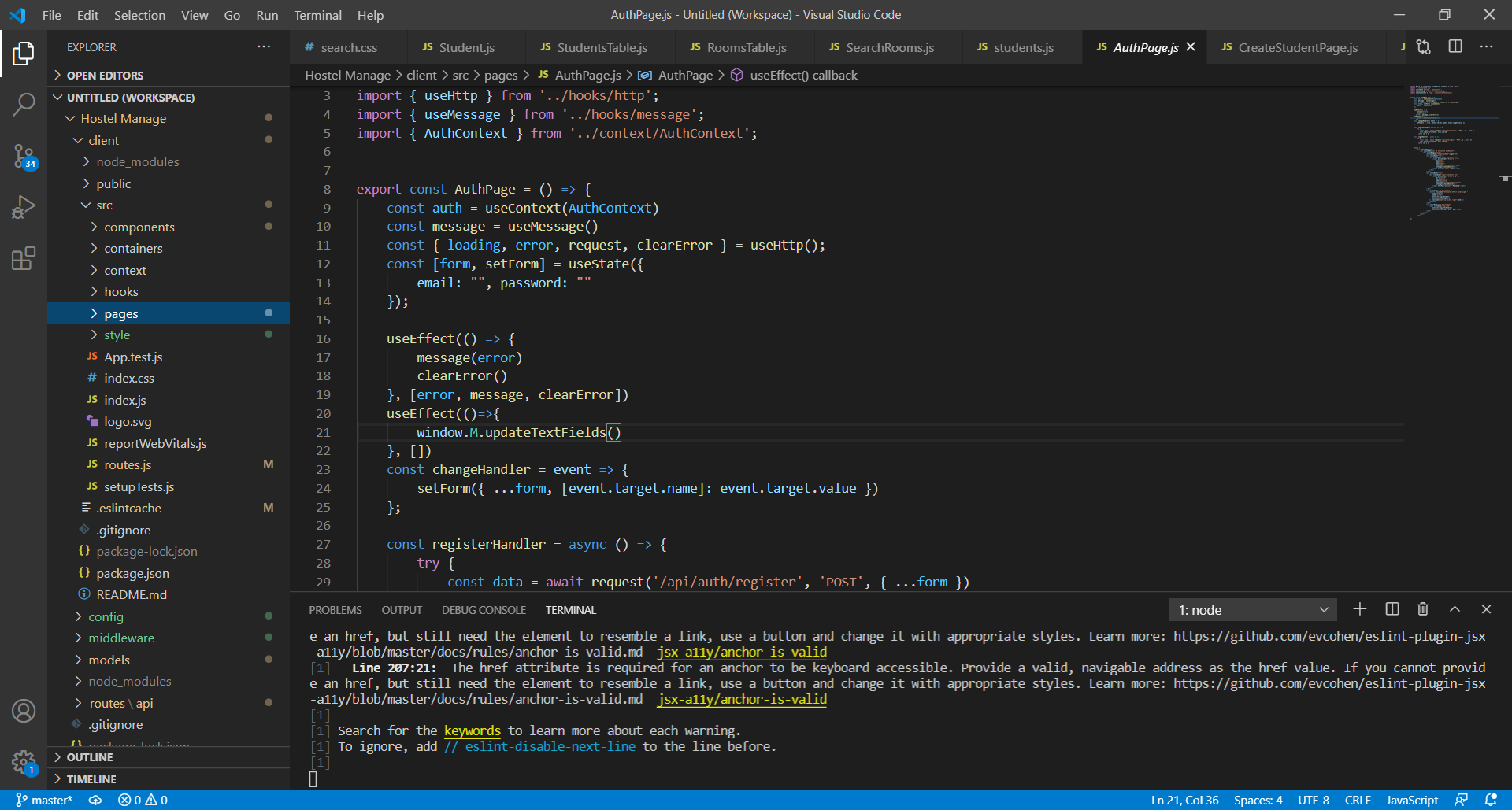


Рисунок 3.2 – Структура пользовательского интерфейса

На рисунке 3.2 представлен пример API приложения для регистрации администраторов.



Рисунок 3.3 – Регистрация администраторов

Из кода можно увидеть, что валидация полей происходит на серверной части, также происходит проверка на существующего администратора, если валидация не проходит, то на клиентское приложение посылается json ответ со статусом 400 и ошибкой, которая произошла, затем эта ошибка появляется на интерфейсе в виде всплывающего уведомления.

Разработка базы данных

В качестве базы данных будет использоваться PostgreSQL, которая позволяет быстро развернуть сервер с базой и легко подключиться к ней.

Для начала необходимо создать таблицы и связи межу ними.

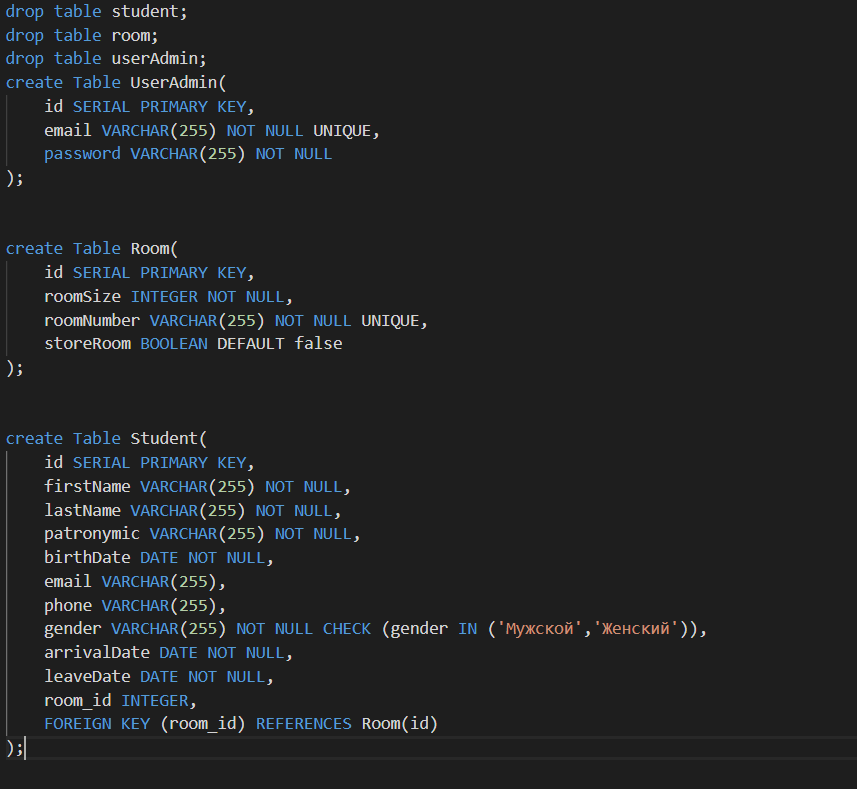


Рисунок 3.6 – Создание базы данных

В качестве первичного ключа в каждой таблице выступает идентификатор, который позволит быстро обращаться к данным таблицы. В таблице «студенты» для поля «пол» установлены ограничения, которые проверяют, чтобы пол был задан только мужским или только женским, все остальные обязательные поля должны быть не пустыми.

# Тестирование

В разделе тестирования будут осуществлены некоторые тесты, которые позволят проверить, все ли работает корректно и насколько предусмотрены ситуации предотвращения возникновения неожиданных ошибок и поломки приложения.

В приложении везде, где происходит взаимодействие клиента и сервера предусмотрены операторы try и catch, которые позволяют отлавливать ошибки, в случае, когда данные ошибки предусмотрены, например, ошибки валидации или некорректности введенных данных, тогда используется специальный хук для вывода всплывающего сообщения об этой ошибке, в случае, когда ошибка непредусмотренная, то в качестве ответа приходит уведомление об ошибке и указание, что произошла серверная ошибка.

Регистрация и авторизация

В качестве первого теста будет выступать проверка регистрации и авторизации пользователей в систему, проверка неправильно введенных значений и правильности полученных уведомлений.

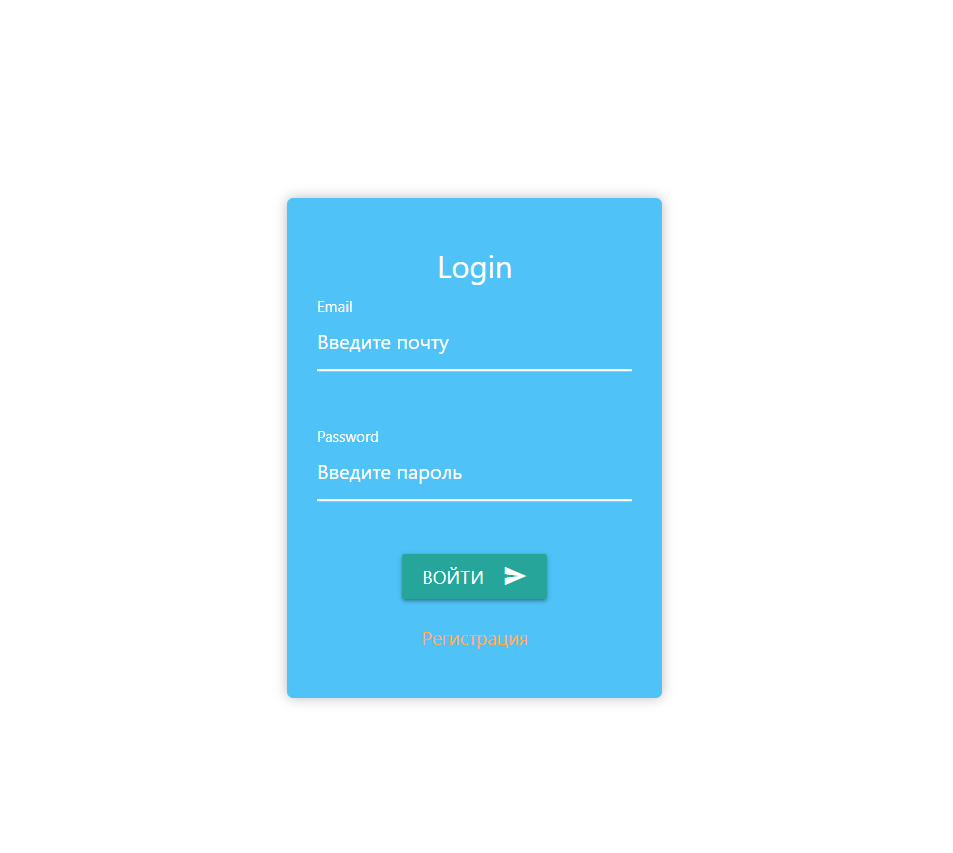


Рисунок 4.1 – Окно авторизации и регистрации

При отсутствии введенных значение и нажатии кнопки войти или регистрация, всплывает уведомление об ошибке, представленной на рисунке 4.2.

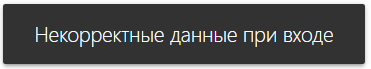


Рисунок 4.2 – Уведомление об ошибки при пустых значениях

Следующим тестом будет проверка на возможность войти в систему под несуществующим пользователем.

Для этого вводим несуществующую почту и случайный пароль и нажимаем кнопку войти.



Рисунок 4.3 – Уведомление, что пользователь не существует

Теперь попробуем ввести правильные данные учетной записи и войти в приложение.

После ввода данных мы сразу оказываемся на главной странице и в правом верхнем углу появляется кнопка выход.

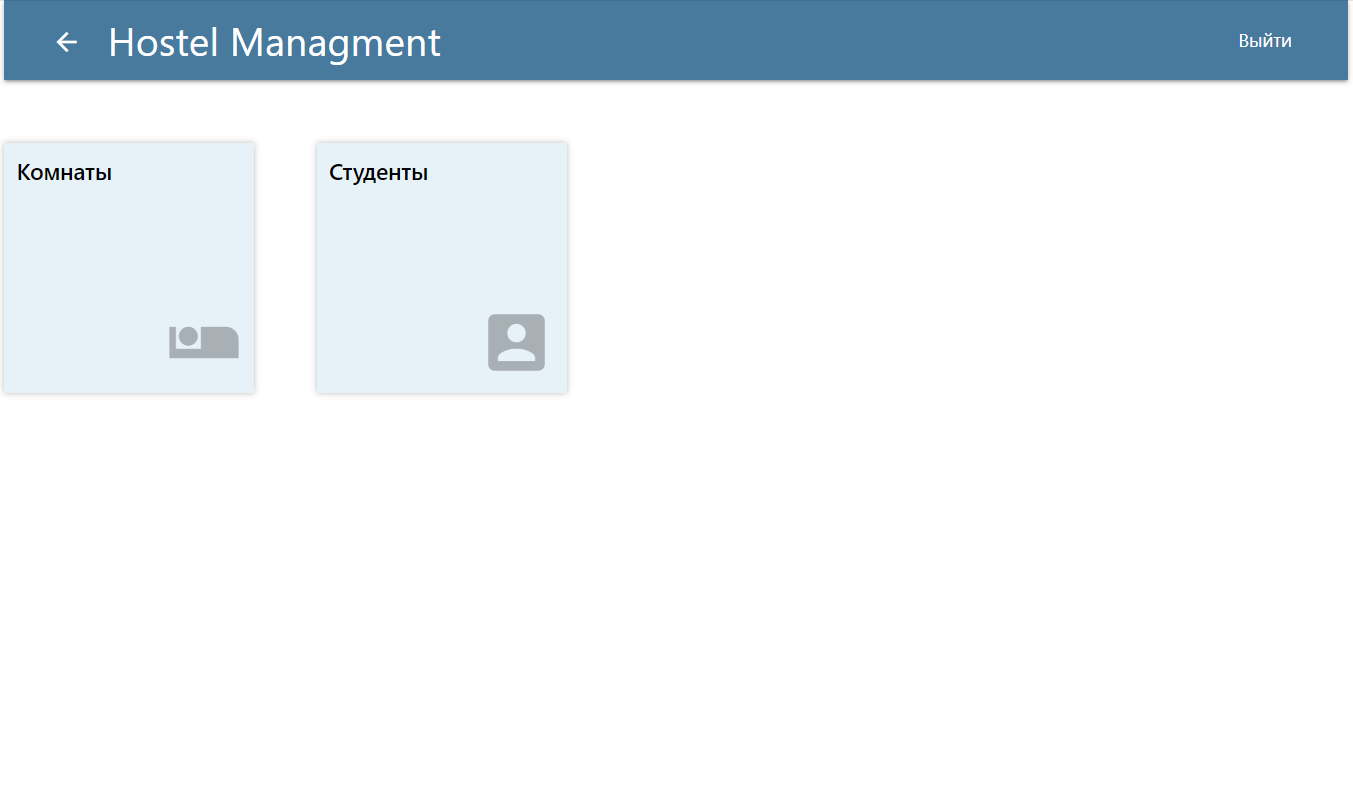


Рисунок 4.4 – Успешный вход в приложение

Также следует отметить, что при попытке перейти на адрес авторизации или регистрации под авторизированным пользователем, перейдет перенаправление на главную страницу, тоже самое произойдет, если нажать кнопку назад сразу после авторизации. Для того, чтобы выйти и перейти к окну авторизации и регистрации, необходимо нажать кнопку выйти.

Изменение информации о комнате

В качестве следующего теста будет рассматриваться возможность введения некорректных данных при изменении информации о комнате.

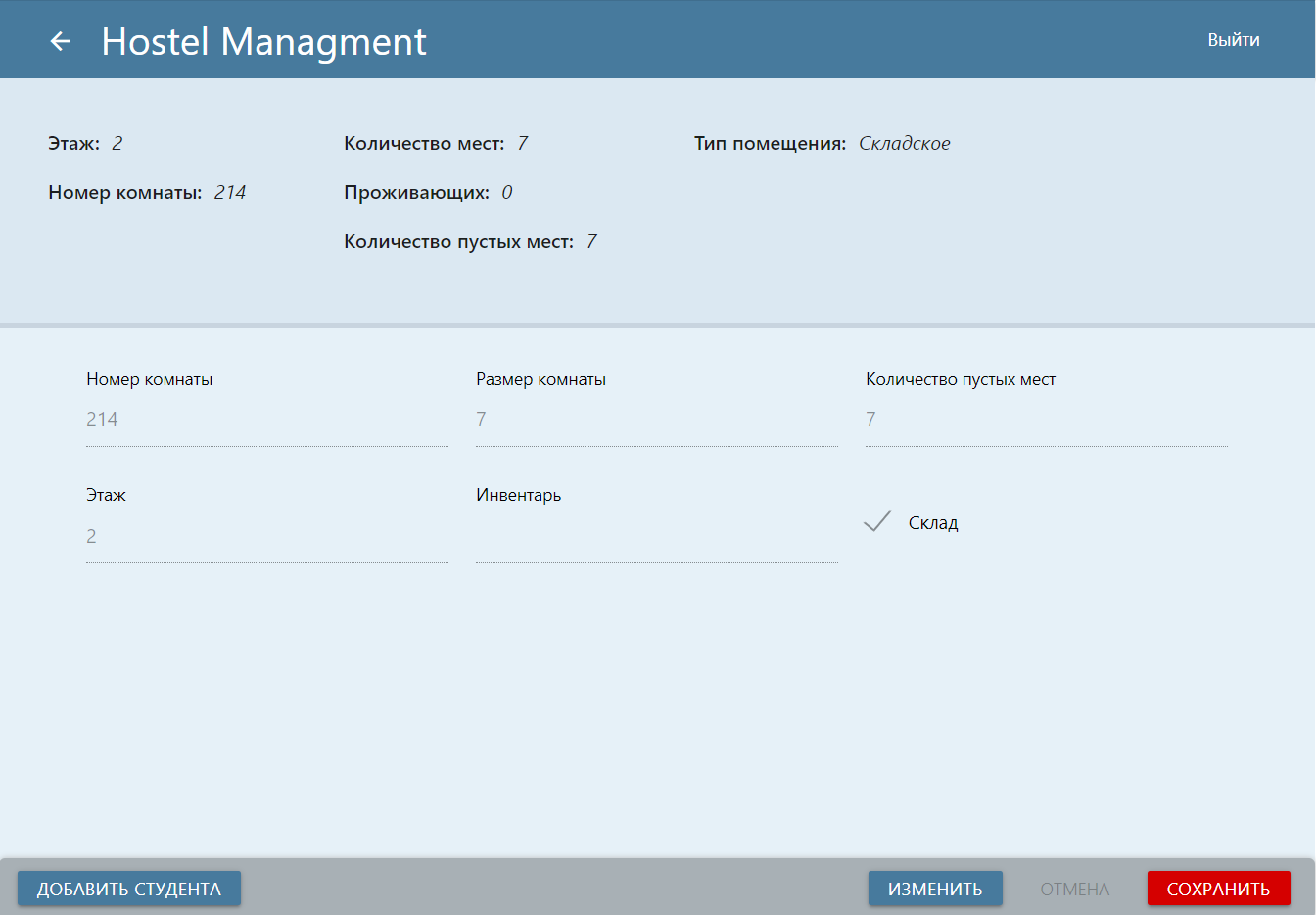


Рисунок 4.5 – Изменение информации о комнате

Для того, чтобы изменить информацию, необходимо нажать кнопку изменить, которая сделает поля для изменения доступными. Ввести текстовое значение в поле размер комнаты невозможно, потому что оно принимает только цифровое значение, поэтому попробуем оставить это поле пустым и нажать сохранить.

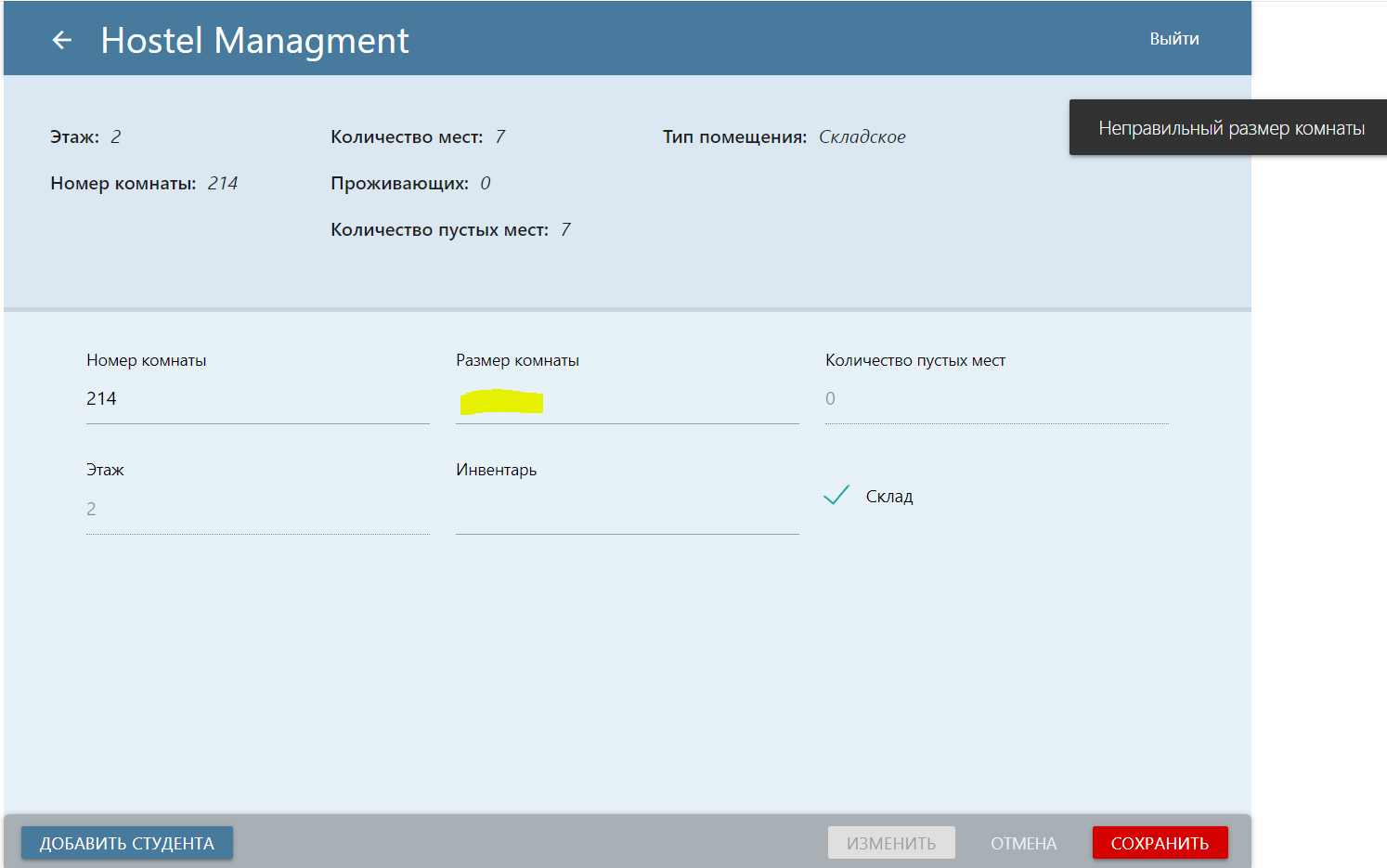


Рисунок 4.6 – Неправильный размер комнаты

Валидация на стороне сервера проходит и приходит ответ, что произошла ошибка, данные не сохраняются.

Теперь попробуем ввести верные значения и обновить информацию о комнате.

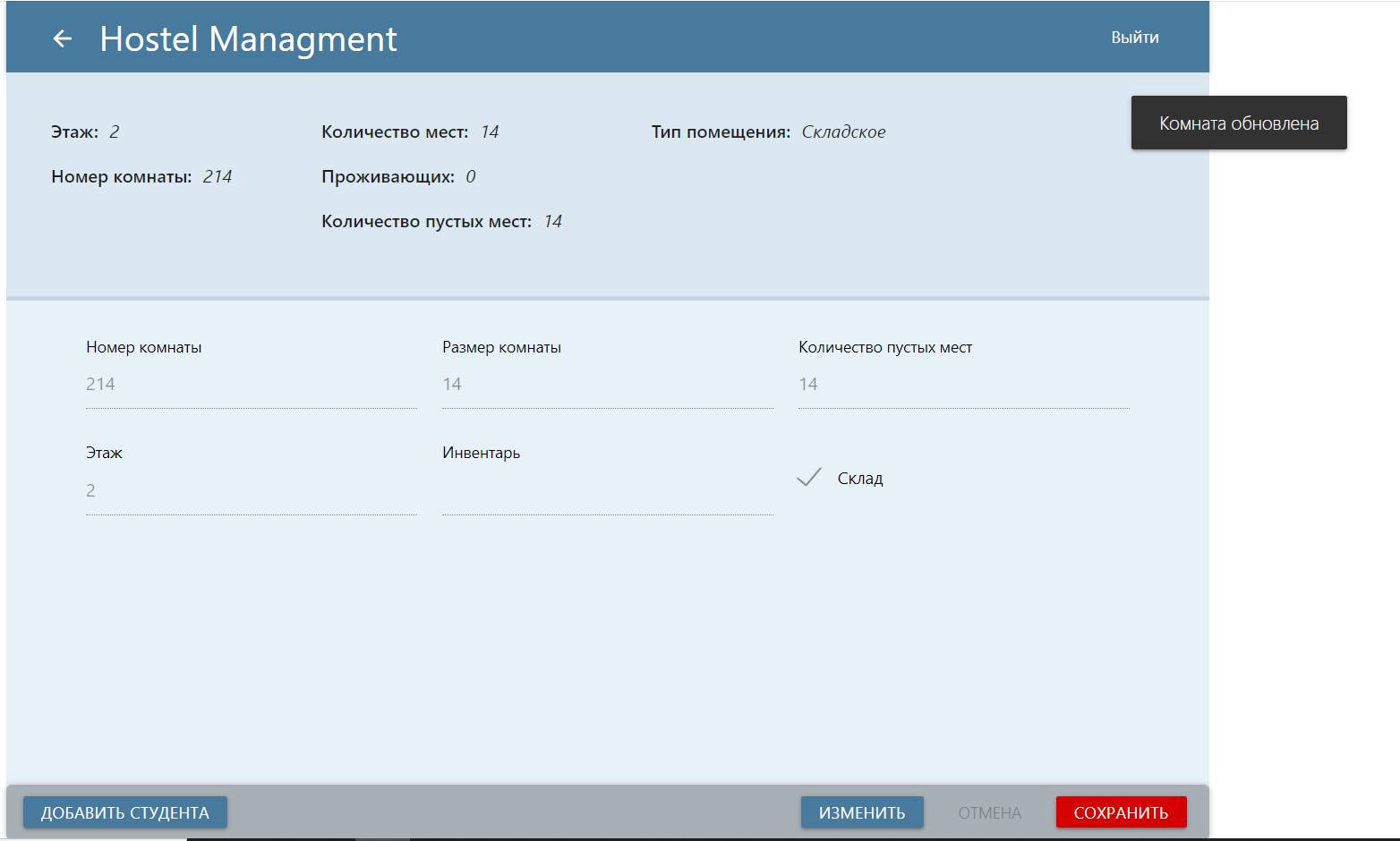


Рисунок 4.7 – Обновление информации о комнате

После успешного обновления приходит уведомление, что комната обновилась и поля ввода становятся неактивными, также в шапке данные обновились в соответствии с введенными.

Для обработки всех результатов применялась технология встраивания middleware, которая позволяет в маршрут между получением запроса и обработкой данных, встроить сколько угодно кода по работе с этими данными, для этого я проверял введенные значение на правильность, затем передавал этот результат вместе с данным в основную функцию обработки запроса и уже там проверял, если данные некорректные, то посылал уведомление. Дальше уже происходила основная бизнес логика по обработке данных и взаимодействию с базой данных, которая проверяла является ли комната складом, могут ли там проживать студенты, сколько свободных мест и возможно ли заселить студента, существует ли такая комната. Приложение предоставляет API, поэтому результаты проверки также доступны при проверке через postman.

# Руководство по развертыванию приложения

Приложение является кроссплатформенным, и я хочу развернуть его с помощью Docker.

Docker — это открытая платформа для разработки, доставки и эксплуатации приложений. Docker разработан для более быстрого выкладывания приложений. С помощью Docker можно отделить приложение от инфраструктуры и обращаться с инфраструктурой как управляемым приложением. Docker помогает выкладывать код быстрее, быстрее тестировать, быстрее выкладывать приложения и уменьшить время между написанием кода и запуска кода. Docker делает это с помощью легковесной платформы контейнерной виртуализации, используя процессы и утилиты, которые помогают управлять и выкладывать ваши приложения.

В своем ядре Docker позволяет запускать практически любое приложение, безопасно изолированное в контейнере. Безопасная изоляция позволяет запускать на одном хосте много контейнеров одновременно. Легковесная природа контейнера, который запускается без дополнительной нагрузки гипервизора, позволяет добиваться больше от возможностей железа компьютера.

Docker использует архитектуру клиент-сервер. Docker клиент общается с демоном Docker, который берет на себя тяжесть создания, запуска, распределения ваших контейнеров. Оба, клиент и сервер могут работать на одной системе, вы можете подключить клиент к удаленному демону Docker. Клиент и сервер общаются через сокет или через Restful API. Docker-демон, т. е. пользователь не взаимодействует с сервером на прямую, а использует для этого клиент. Docker-клиент – это программа Docker, т. е. главный интерфейс к Docker. Она получает команды от пользователя и взаимодействует с Docker-демоном.

FROM node:alpine

WORKDIR '/app'

COPY package.json .

RUN npm install

COPY . .

CMD ["npm","start"]

Листинг 5.1 – Dockerfile серверной части проекта

Здесь указано, что мы создаем рабочую область и копируем на Docker package.json, который используется для загрузки npm пакетов, которые использовались в приложении, а после этого запускаем сервер командой «npm start».

После этого запускаем созданный image файл на Docker.

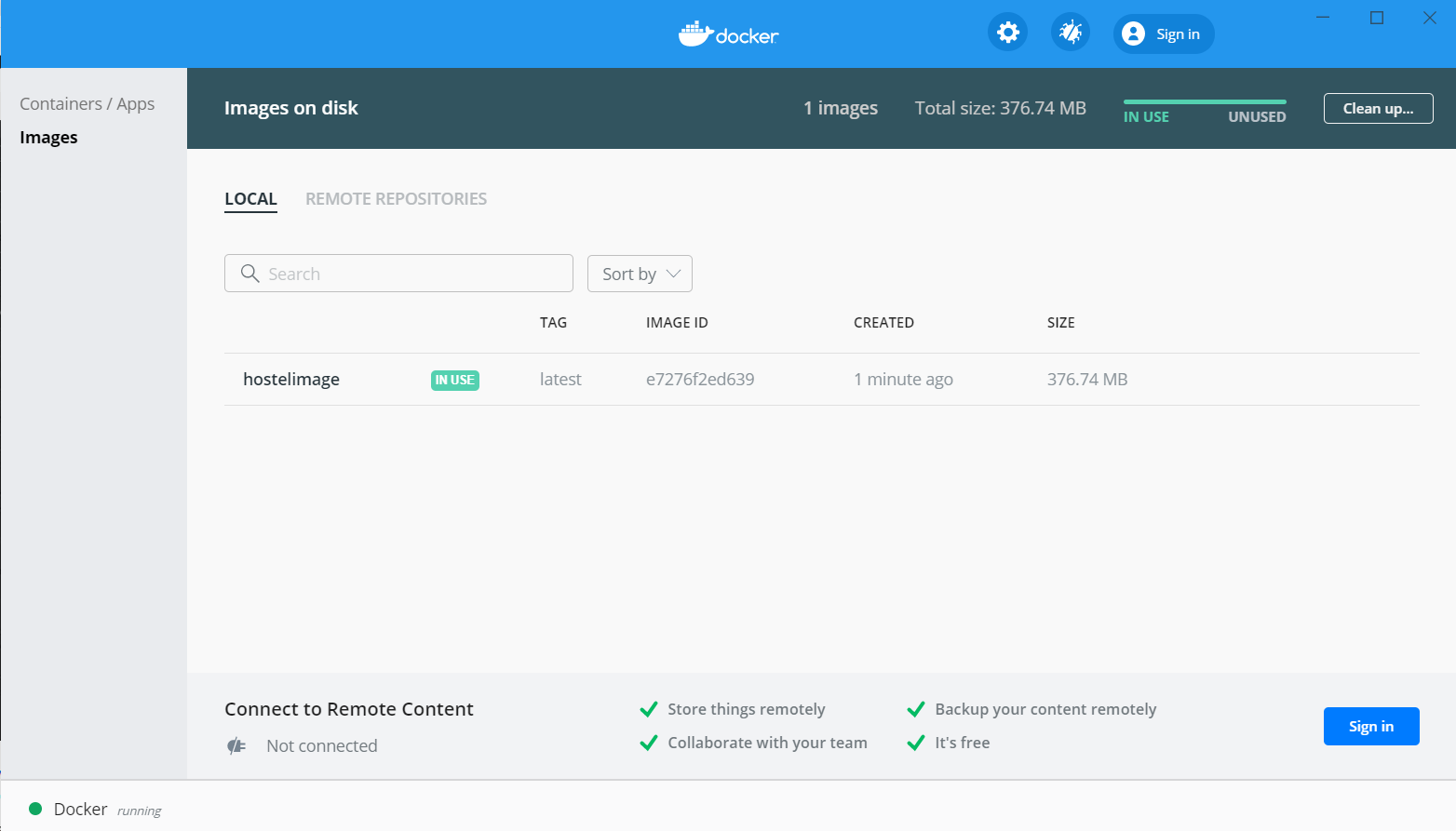


Рисунок 5.2 – Запуск image на Docker

После этого открываем консоль image на Docker и запускаем приложение.

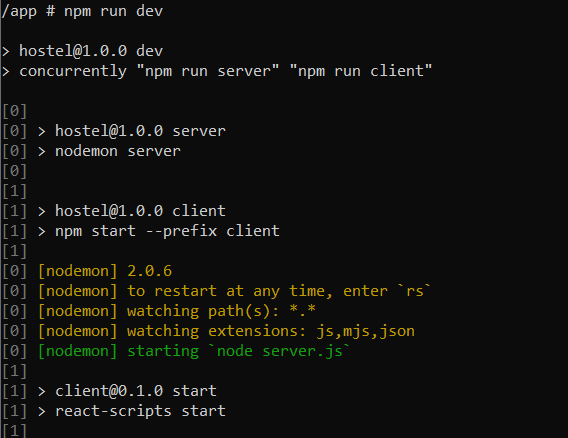


Рисунок 5.3 – Запуск приложения на Docker

После запуска приложение доступно по адресу в браузере. Теперь его можно использовать.

# Заключение

В ходе выполнения работы были проанализированы аналоги веб-приложения и десктоп-приложения для администрирования отелей и хостелов, что позволило произвести оценку существующих проектов на рынке, также можно было сравнить интерфейс и скорость работы, перегруженность приложения различным функционалом, который не всех требовался и который невозможно было скрыть. Все приложения являлись платными и стоили очень много за использование в месяц, даже простое и очень старое десктоп-приложение предоставляло пробный период всего на 60 открытий и требовало затем заплатить большую сумму для доступа к полному функционалу. Плюсом является то, что данные приложения разрабатывала большая команда опытных специалистов разных областей, за разработку и дизайн интерфейса отвечал дизайнер, за разработку сервера и базы данных отвечали отдельные специалисты, что позволило создать качественный и конкурентно-способный продукт.

Целью данной работы не являлось создать что-то, что могло бы конкурировать с существующими продуктами, а разработать свое приложение, чтобы оценить степень трудоемкости, а также придумать новые способы взаимодействия или интерфейс. В ходе выполнения работы было обнаружено, что приложение для администрирования общежития должно быть построено с использованием реляционной базы данных т. к. в приложении присутствуют связи между сущностями студентов и комнат. Изначально приложение разрабатывалось с использованием NoSQL базы данных, что сильно затормозило процесс т. к. не было возможности настроить связи между сущностями и обращаться к ним, потом приложение было переписано под PostgreSQL, которая является реляционной базой данных и уже после этого я смог завершить разработку приложения.

Также, немаловажным фактором является выбор библиотеки или фреймворка для разработки клиентской части, в ходе исследования и оценки скорости работы и предоставляемого функционала, была выбрана библиотека React, но данная библиотека сильно нагружена ненужными компонентами, так что при разработке приложения с простым интерфейсом следует обратить внимание на аналоги. Конечно, несомненным плюсом является скорость обработки данных и обновления представления, что пересоздаются только те части интерфейса в которых произошли изменения, но использование хуков, компонентов, состояний значительно усложнило структуру приложения, а любая сложная структура требует тщательного проектирования архитектуры и большой опыт в создании качественного и независимого интерфейса, чего не удалось мне добиться, т. к. это был первый опыт с данной библиотекой.

В плюсы хочу выделить серверную часть приложения Node.js. На практике выбор Node.js оказался одним из лучших серверов, что я встречал. Большая скорость работы, выполнение асинхронных процедур, простота создания архитектуры веб-сервера, которая предоставляет API для взаимодействия с данными и маршрутизацию и интеграцию со множеством сервисов.

# Список литературы

Preno – Simple Hotel Property Management Software [Электронный ресурс] / Preno 2020 – Режим доступа: https://prenohq.com/ – Дата доступа 21.11.2020

The Booking Factory: All-In-One Hotel Management System [Электронный ресурс] / Booking Factory 2014 – 2018 – Режим доступа: <https://thebookingfactory.com/> – Дата доступа 21.11.2020

Hotel reservation system | Hotel software | Sirvoy [Электронный ресурс] / Frappe – Copyright © 2008-2020 Sirvoy Ltd – Режим доступа: https://sirvoy.com/ – Дата доступа 21.11.2020

GitHub [Электронный ресурс] / GitHub Inc. – Сан-Франциско, 2020 – Режим доступа: https://github.com/svotoq– Дата доступа 21.11.2020

ПРИЛОЖЕНИЕ А

Сервер

const express = require('express');

const bodyParser = require('body-parser');

const config = require('config')

//Run the app

const app = express();

//use bodyParser

app.use(bodyParser.json());

//Define api routes

app.use("/api/auth", require("./routes/api/auth"));

app.use("/api/rooms", require("./routes/api/rooms"));

app.use("/api/students", require("./routes/api/students"));

const PORT = config.get('port') || 5000;

async function start() {

try {

app.listen(PORT, () => {

console.log("App has been started on port ${PORT}...");

});

} catch (e) {

console.log("Server Error", e.message);

process.exit(1);

}

}

start();

Обработка запросов к студентам

const express = require('express');

const router = express.Router();

const Student = require('../../models/Student');

const { check, validationResult } = require('express-validator');

const Room = require('../../models/Room');

// GET /api/students/

router.get("/", async (req, res) => {

try {

const students = await Student.getStudents()

res.json(students)

}

catch (e) {

res.status(500).json({ message: e.message })

}

});

// GET /api/students/:id

router.get("/:id", async (req, res) => {

try {

const student = await Student.getStudent(req.params.id)

res.json(student)

}

catch (e) {

res.status(500).json({ message: e.message })

}

});

// POST /api/students/

router.post("/",

[

check('firstname', 'Некорректное имя').isString().isLength({ min: 1 }),

check('lastname', 'Некорректная фамилия').isString().notEmpty(),

check('patronymic', 'Некорректное отчество').isString().notEmpty(),

check('birthdate', 'Некорректная дата рождения').notEmpty(),

check('gender', 'Некорректный пол').isString().notEmpty(),

check('arrivaldate', 'Некорректная дата заселения').notEmpty(),

check('leavedate', 'Некорректная дата выселения').notEmpty(),

check('room\_id', 'Укажите комнату').notEmpty()

], async (req, res) => {

try {

const errors = validationResult(req)

if (!errors.isEmpty()) {

return res.status(400).json({

errors: errors.array(),

message: "Некорректные данные при добавлении студента"

})

}

const room = await Room.getRoom(req.body.room\_id)

if (room.roomsize - room.students.length === 0) {

return res.status(400).json({ message: `Комната ${room.roomnumber} заполнена` })

}

if (room.storeroom) {

return res.status(400).json({ message: `Комната ${room.roomnumber} является складским помещением` })

}

const student = await Student.create(req.body)

return res.status(201).json({ message: 'Студент успешно создан' })

}

catch (e) {

res.status(500).json({ message: e.message })

}

});

// POST /api/students/search

router.post("/search", async (req, res) => {

try {

let searchParams = { ...req.body }

delete searchParams.roomnumber

let students = await Student.find(searchParams)

searchParams = { ...req.body }

if (searchParams.roomnumber) {

students = students.filter(student => student.room.roomnumber === searchParams.roomnumber)

}

res.json(students)

}

catch (e) {

res.status(500).json({ message: e.message })

}

});

// PUT /api/students/

router.put("/:id",

[

check('firstname', 'Некорректное имя').isString().isLength({ min: 1 }),

check('lastname', 'Некорректная фамилия').isString().notEmpty(),

check('patronymic', 'Некорректное отчество').isString().notEmpty(),

check('birthdate', 'Некорректная дата рождения').notEmpty(),

check('gender', 'Некорректный пол').isString().notEmpty(),

check('arrivaldate', 'Некорректная дата заселения').notEmpty(),

check('leavedate', 'Некорректная дата выселения').notEmpty(),

check('room\_id', 'Укажите комнату').notEmpty()

], async (req, res) => {

try {

const errors = validationResult(req)

if (!errors.isEmpty()) {

return res.status(400).json({

errors: errors.array(),

message: "Некорректные данные при добавлении студента"

})

}

const existsStudent = await Student.getStudent(req.params.id)

if(!existsStudent) {

return res.status(500).json({message: 'Студент не существует'})

}

if (existsStudent.room\_id != req.body.room\_id) {

const room = await Room.getRoom(req.body.room\_id)

if (room.roomsize - room.students.length === 0) {

return res.status(400).json({ message: `Комната ${room.roomnumber} заполнена` })

}

if (room.storeroom) {

return res.status(400).json({ message: `Комната ${room.roomnumber} является складским помещением` })

}

}

const student = await Student.update(req.body, req.params.id)

res.json(student)

}

catch (e) {

res.status(500).json({ message: e.message })

}

});

ПРИЛОЖЕНИЕ Б

Страница списка комнат

import React, { useCallback, useContext, useEffect, useState } from 'react'

import { useHttp } from '../../hooks/http';

import { useMessage } from '../../hooks/message';

import { RoomsTable } from './RoomsTable'

import '../../style/search.css'

import 'materialize-css'

export const SearchRooms = () => {

const { loading, error, request, clearError } = useHttp();

const emptyForm = {

roomnumber: "", roomsize: "",

emptyplaces: "", floornumber: "",

inventory: ""

}

const emptyChkbks = {

emptyroom: false, storeroom: false

}

const [form, setForm] = useState(emptyForm)

const [checkbs, setCheckbs] = useState(emptyChkbks)

const [rooms, setRooms] = useState([])

const checkbsChangeHandler = event => {

setCheckbs({ ...checkbs, [event.target.name]: event.target.checked })

}

const changeHandler = event => {

setForm({ ...form, [event.target.name]: event.target.value })

}

const searchHandler = async () => {

try {

const searchParams = { ...form, ...checkbs }

Object.keys(searchParams).forEach((key) => (searchParams[key] == "") && delete searchParams[key])

const data = await request('/api/rooms/search', 'POST', { ...searchParams })

setRooms({})

setRooms(data)

} catch (e) {

console.log(e.message)

}

}

const resetHandler = () => {

setForm(emptyForm)

setCheckbs(emptyChkbks)

document.querySelector("#storeroom").checked = false;

document.querySelector("#emptyroom").checked = false;

fetchRooms()

}

const fetchRooms = useCallback(async () => {

try {

const fetched = await request('/api/rooms', 'GET')

setRooms({})

setRooms(fetched)

} catch (e) { }

}, [request])

useEffect(() => {

fetchRooms()

}, [fetchRooms])

return (

<div>

<div className="search-container search-rooms">

<div className="row">

<div className="search-input-field col s4">

<label htmlFor="roomnumber">Номер комнаты</label>

<input id="roomnumber"

type="text"

name="roomnumber"

onChange={changeHandler}

value={form.roomnumber}/>

</div>

<div className="search-input-field col s4">

<label htmlFor="roomsize">Размер комнаты</label>

<input id="roomsize"

type="number"

name="roomsize"

value={form.roomsize}

onChange={changeHandler}/>

</div>

<div className="search-input-field col s4">

<label htmlFor="emptyplaces">Количество пустых мест</label>

<input id="emptyplaces"

type="number"

name="emptyplaces"

value={form.emptyplaces}

onChange={changeHandler}/>

</div>

</div>

<div className="row">

<div className="search-input-field col s4">

<label htmlFor="floornumber">Этаж</label>

<input id="floornumber"

type="number"

name="floornumber"

value={form.floornumber}

onChange={changeHandler}

className="validate" />

</div>

<div className="search-input-field col s4">

<label htmlFor="inventory">Инвентарь</label>

<input id="inventory"

type="text"

name="inventory"

value={form.inventory}

onChange={changeHandler}

className="validate" />

</div>

<div className="checkbox-containter col s4">

<div className="col s6">

<label>

<input type="checkbox"

name="emptyroom"

id="emptyroom"

onChange={checkbsChangeHandler}

/>

<span>Свободная</span>

</label>

</div>

<div className="col s6">

<label>

<input type="checkbox"

name="storeroom"

id="storeroom"

onChange={checkbsChangeHandler}

/>

<span>Склад</span>

</label>

</div>

</div>

</div>

<div className="row bottom-row">

<a className="waves-effect waves-light btn-flat reset-button"

onClick={resetHandler}>Сбросить</a>

<a className="waves-effect waves-light btn red accent-4 search-button"

onClick={searchHandler}>GO</a>

</div>

</div>

{<RoomsTable rooms={rooms} />}

</div>

)

}