МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И. Сатпаева

Институт информационных и телекоммуникационных технологий

Кафедра кибербезопасность, обработка и хранение информации

Жұмәлі Сымбат

Разработка информационной системы гостиничного комплекса

ДИПЛОМНАЯ РАБОТА

Специальность 5В070300 – Информационные системы

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И. Сатпаева

Институт информационных и телекоммуникационных технологий

Кафедра кибербезопасность, обработка и хранение информации

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
КБОиХИ, канд.техн.наук,
ассистент-профессор.
_____Н.А. Сейлова
«_16_» мая 2019 г.

ДИПЛОМНАЯ РАБОТА

На тему: «Разработка информационной системы гостиничного комплекса»

Специальность 5В070300 – Информационные системы

Выполнил С.Жұмәлі

Рецензент,
PhD, ассистент проф. СДУ
_____ О. А. Баймуратов
«_15_» мая 2019 г.

Научный руководитель,
PhD, сениор лектор
_____О.В. Киселева
«_14_» мая 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И. Сатпаева

Институт информационных и телекоммуникационных технологий

Кафедра кибербезопасность, обработка и хранение информации

5В070300 – Информационные системы

УТВЕРЖДАЮ
Заведующий кафедрой
КБОиХИ, канд.техн.наук,
ассистент-профессор.
_____Н.А. Сейлова
«_16_» мая 2019 г.

ЗАДАНИЕ
на выполнение дипломной работы

Обучающемуся Жұмәлі Сымбат
Тема: Разработка информационной системы гостиничного комплекса.
Утверждена приказом Ректора Университета №116 2-б от 16.10.2018 г.
Срок сдачи законченной работы «_17_» _мая_ 2019 г.
Исходные данные к дипломной работе: результат литературного обзора современного состояния по данной теме, сбор теоретического материала.
Краткое содержание дипломной работы:
а) анализ и обзор гостеприимства;
б) разработка веб-приложения;
Перечень графического материала (с точным указанием обязательных чертежей): представлены _16_ слайдов презентации работы.
Рекомендуемая основная литература: из _25_ наименований _15_

ГРАФИК
подготовки дипломной работы

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю и консультантам	Примечание
Обзор тенденции гостеприимства	10.01.2019 – 08.03.2019	
Модели и служба функционирования системы	05.02.2019 – 10.03.2019	
Информационное и программное обеспечение	11.03.2019 – 28.04.2019	

Подписи
консультантов и нормоконтролера на законченную дипломную работу с указанием относящихся к ним разделов работы

Наименование разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Разработка ПО	м.т.н М.Б. Бауржан	13.05.18г	
Нормоконтроллер	Сениор-лектор, PhD О. В. Киселева	13.05.18	

Научный руководитель _____ Киселева О. В.

Задание принял к исполнению обучающийся _____ Жұмәлі Сымбат.

Дата "08" 01 20 19 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
СӘТБАЕВ УНИВЕРСИТЕТІ

РЕЦЕНЗИЯ

на дипломную работу
Жұмәлі Сымбат Қайратқызы
5В070300 - Информационные системы
На тему: Разработка информационной системы гостиничного комплекса

Выполнено:
а) графическая часть на 15 листах
б) пояснительная записка на 30 страницах

ЗАМЕЧАНИЯ К РАБОТЕ

- имеются грамматические и стилистические ошибки;
- ссылки на источники информации не приведены полностью;
- не приведено подробное описание библиотек при разработке программы/системы;
- структура таблиц баз данных не представлена в дипломной работе;
- описание и требования для эксплуатации разработанной программы не приведены.

Оценка работы

Работа частично выполнена на должном уровне и имеет практическую значимость. Имеющиеся недостатки поправимы за короткое время-до защиты дипломной работы. Рекомендую автора работы Жұмәлі Сымбат Қайратқызы для присвоения степени бакалавра.

Оценка за дипломную работу (75 %) - хорошо.

Рецензент
Ассоц. проф., PhD.
(должность, уч. степень, звание)
_____ Баймуратов О.А.
(подпись)
«15» 05 2019 г.

Ф КазНИТУ 706-17. Рецензия

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
СӘТБАЕВ УНИВЕРСИТЕТІ

ОТЗЫВ

НАУЧНОГО РУКОВОДИТЕЛЯ

на дипломную работу
(наименование вида работы)
Жұмәлі Сымбат Қайратқызы
(Ф.И.О. обучающегося)
5В070300 – Информационные системы
(шифр и наименование специальности)

Тема: Разработка информационной системы гостиничного комплекса

Объектом своего исследования Жұмәлі С.Қ. выбрала веб-приложение для гостиничного комплекса. Сегодня гостиничный бизнес - одна из наиболее перспективных и быстроразвивающихся отраслей, приносящей по всему миру многомиллионные прибыли. С каждым годом растет количество гостиниц, как в нашей стране, так и за рубежом. Поэтому тема разработки программного веб-приложения «Гостиница» в настоящее время является актуальной, так как гостиничный бизнес активно развивается, разрастаются сети отелей, создавая на рынке конкуренцию.

В «информационной системе гостиничного комплекса» представлены услуги размещения, питания и прочие услуги, предоставляемые гостиничным предприятием. В ходе выполнения данной дипломной работы необходимо разработать информационную базу данных для гостиницы, которая поможет любому пользователю легко найти нужную информацию о подходящей ему категории гостиницы. Приложение разработано корректно, с использованием языка Python.

В дипломной работе рассматриваются инструменты для создания мобильного приложения, такие как Python, фреймворк Django, текстовый редактор Atom, СУБД SQLite. Во второй главе ведется обоснование выбора программного обеспечения, в третьей главе ведется разработка приложения.

В процессе написания дипломной работы Жұмәлі С.Қ. показала себя самостоятельным, хорошо подготовленным специалистом. Также она проявила высокий уровень знаний.

Рекомендую дипломную работу, выполненную Жұмәлі С.Қ., к защите. Считаю, что она заслуживает присвоения академической степени бакалавра по специальности 5В070300 – Информационные системы.

Научный руководитель
PhD, сеньор-лектор
(должность, уч. степень, звание)
_____Киселева О.В.
(подпись)
«16» мая 2019 г.

Протокол анализа Отчета подобия

заведующего кафедрой / начальника структурного подразделения

Заведующий кафедрой / начальник структурного подразделения заявляет, что ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Жұмәлі Сымбат

Название: Разработка информационной системы гостиничного комплекса

Координатор: Ольга Киселева

Коэффициент подобия 1:16,7

Коэффициент подобия 2:14

Тревога:2

После анализа отчета подобия заведующий кафедрой / начальник структурного подразделения констатирует следующее:

✓ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, работа признается самостоятельной и допускается к защите;
 обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
 обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, работа не допускается к защите.

Обоснование:

Заимствование добросовестное

Дата 16.05.19г

Подпись заведующего кафедрой /

начальника структурного подразделения

Окончательное решение в отношении допуска к защите, включая обоснование:

Корпуса жіберіці

Дата 16.05.19г

Подпись заведующего кафедрой

начальника структурного подразделения

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Жұмәлі Сымбат

Название: Разработка информационной системы гостиничного комплекса

Координатор:Ольга Киселева

Коэффициент подобия 1:16,7

Коэффициент подобия 2:14

Тревога:2

После анализа Отчета подобия констатирую следующее:

- ☑ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;

- ☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

- ☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

Взаимствования являются добросовестными, и не обладают признаками плагиата

15.05.19 О.К.

Дата *Подпись Научного руководителя*

Краткий отчет

Университет:	Satbayev University
Название:	Разработка информационной системы гостиничного комплекса
Автор:	Жұмәлі Сымбат
Координатор:	Ольга Киселева
Дата отчета:	2019-05-13 11:16:57
Коэффициент подобия № 1:	**16,7%**
Коэффициент подобия № 2:	**14,0%**
Длина фразы для коэффициента подобия № 2:	**25**
Количество слов:	3 171
Число знаков:	24 191
Адреса пропущенные при проверке:	
Количество завершенных проверок:	56

! **К вашему сведению, некоторые слова в этом документе содержат буквы из других алфавитов. Возможно - это попытка скрыть позаимствованный текст. Документ был проверен путем замещения этих букв латинским эквивалентом. Пожалуйста, уделите особое внимание этим частям отчета. Они выделены соответственно.**
Количество выделенных слов 2

Самые длинные фрагменты, определеные, как подобные

№	Название, имя автора или адрес гиперссылки *(Название базы данных)*	Количество Автородинаковых слов

1	URL_ https://knowledge.allbest.ru/programming/3c0a65635a2ac78b5d43b88521316c26_0.html		365
2	URL_ https://knowledge.allbest.ru/programming/3c0a65635a2ac78b5d43b88521316c26_0.html		41
3	URL_ https://knowledge.allbest.ru/programming/3c0a65635a2ac78b5d43b88521316c26_0.html		38
4	Документ из базы НЭУ 525235da-4c3c-4aef-b025-0711c0a80d1d.docx *NARXOZ (NEU) (Information Technology Center)*	na	19
5	URL_ https://knowledge.allbest.ru/programming/3c0a65635a2ac78b5d43b88521316c26_0.html		15
6	Документ из базы НЭУ 51dbbebd-6f28-4773-881e-3910c0a80d1d.docx *NARXOZ (NEU) (Information Technology Center)*	na	14
7	Лигай Э.docx *Kazakh University of Economics, Finance and International Trade (KUEFIT) (ФАКУЛЬТЕТ ПОСЛЕВУЗОВСКОГО ОБРАЗОВАНИЯ)*	Лигай Э.	13
8	Документ из базы НЭУ 525235da-4c3c-4aef-b025-0711c0a80d1d.docx *NARXOZ (NEU) (Information Technology Center)*	na	9
9	Документ из базы НЭУ 525235da-4c3c-4aef-b025-0711c0a80d1d.docx *NARXOZ (NEU) (Information Technology Center)*	na	6
10	Документ из базы НЭУ 525235da-4c3c-4aef-b025-0711c0a80d1d.docx *NARXOZ (NEU) (Information Technology Center)*	na	6

>> Документы,содержащие подобные фрагменты: Из домашней базы данных

Не обнаружено каких-либо заимствований

>> Документы,содержащие подобные фрагменты: Из внешних баз данных

Документы, выделенные жирным шрифтом, содержат фрагменты потенциального плагиата, то есть превышающие лимит в длине коэффициента подобия № 2

№	Название *(Название базы данных)*	Автор	Количество одинаковых слов (количество фрагментов)
1	Документ из базы НЭУ 525235da-4c3c-4aef-b025-0711c0a80d1d.docx *NARXOZ (NEU) (Information Technology Center)*	na	40 (4)
2	Документ из базы НЭУ 51dbbebd-6f28-4773-881e-3910c0a80d1d.docx *NARXOZ (NEU) (Information Technology Center)*	na	14 (1)
3	Лигай Э.docx *Kazakh University of Economics, Finance and International Trade (KUEFIT) (ФАКУЛЬТЕТ ПОСЛЕВУЗОВСКОГО ОБРАЗОВАНИЯ)*	Лигай Э.	13 (1)

>> Документы,содержащие подобные фрагменты: Из интернета

Документы, выделенные жирным шрифтом, содержат фрагменты потенциального плагиата, то есть превышающие лимит в длине коэффициента подобия № 2

№	Источник гиперссылки	Количество одинаковых слов (количество фрагментов)
1	URL_ https://knowledge.allbest.ru/programming/3c0a65635a2ac78b5d43b88521316c26_0.html	464 (5)

Copyright © Plagiat.pl 2002-2019

АҢДАТПА

Дипломдық жұмыстын атауы: «қонақ үйдің ақпараттық жүйесін әзірлеу».Бұл дипломдық жұмыс "қонақ үй кешенінің"Web-қосымшасын әзірлеу болып табылады. "Қонақ үй кешенінің ақпараттық жүйесінде" қонақ үй кәсіпорны ұсынатын орналастыру, тамақтану және басқа да қызметтер көрсетілген.

Web-қосымшаны әзірлеу Python 3 бағдарламалау тілінің көмегімен жүзеге асырылды – жалпы мақсаттағы жоғары деңгейлі бағдарламалау тілі және Django фреймворк. Сонымен қатар, Python, Django, HTML кітапханалары пайдаланылды.Қонақ үй жұмысының негізгі тұжырымдамасы қарастырылды.Дипломдық жұмыстың бірегейлігі-әлемнің кез келген нүктесінен клиент нөмірлерді онлайн брондауды жүзеге асыра алады.

АННОТАЦИЯ

Название дипломной работы: «Разработка информационной системы гостиничного комплекса». Данная дипломная работа представляет собой разработку Web-приложения «Гостиничного комплекса». В «информационной системе гостиничного комплекса» представлены услуги размещения, питания и прочие услуги, предоставляемые гостиничным предприятием.

Разработка web-приложения осуществлена при помощи языка программирования Python3 – высокоуровневый язык программирования общего назначения и фреймворка Django. При разработке также были использованы библиотеки Python, Django, HTML.Были рассмотрены основные концепции работы гостиницы.Уникальность дипломной работы в том что из любой точки мира клиент может совершить онлайн бронирование номеров.

ANNOTATION

The name of the diploma work: «development of the information system of the hotel». This thesis is a development of the Web application of the “Hotel Complex”. The “information system of the hotel complex” includes accommodation, catering and other services provided by the hotel enterprise.

The development of a web application was carried out using the programming language Python3, a high-level general-purpose programming language and the Django framework. The development also used the library Python, Django, HTML. The main concepts of the hotel were considered. The thesis of the thesis is that from anywhere in the world a client can make online room reservations.

СОДЕРЖАНИЕ

	ВВЕДЕНИЕ	7
1	Анализ и обзор современных тенденции развития индустрии гостеприимства	8
1.1	Виды веб-приложений	9
1.2	Архитектура клиент-сервер	10
1.3	Анализ существующих технологий создания веб-приложений	11
1.4	Постановка задач дипломной работы	15
2	Модели и службы функционирования системы	15
2.1	Функциональная модель системы	16
2.2	Обобщенный алгоритм функционирования системы	16
2.3	Информационное обеспечение системы	17
2.4	Обоснование выбора СУБД	19
2.5	Разработка структуры базы данных	22
3	Информационное и программное обеспечение	24
3.1	Структура программного обеспечения	24
3.2	Системное программное обеспечение	25
3.3	Инструментальное программное обеспечение	26
3.4	Прикладное программное обеспечение	27
3.5	Функциональное назначение	28
3.6	Работа и апробация веб-приложения	28
	ЗАКЛЮЧЕНИЕ	30
	СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	31
	Приложение А	32
	Приложение Б	48

ВВЕДЕНИЕ

Всем нам часто приходится путешествовать. И не важно, отдых это, поездка в гости, или же просто командировка, останавливаться где-то нужно, даже, если срок пребывания в другом городе небольшой. Конечно, именно для таких случаев и были придуманы гостиницы, чтобы человек, оказавшейся в другом городе, мог без проблем найти место для ночлега, не беспокоя своих коллег или друзей. Сегодня гостиничный бизнес - одна из наиболее перспективных и быстроразвивающихся отраслей, приносящей по всему миру многомиллионные прибыли. С каждым годом растет количество гостиниц, как в нашей стране, так и за рубежом. Поэтому тема разработки программного веб-приложения «Гостиница» в настоящее время является актуальной, так как гостиничный бизнес активно развивается, разрастаются сети отелей, создавая на рынке конкуренцию.

В ходе выполнения данной дипломной работы необходимо разработать информационную базу данных для гостиницы, которая поможет любому пользователю легко найти нужную информацию о подходящей ему категории гостиницы. Задачей администратора гостиницы является отслеживание финансовой стороны работы гостиницы. Его деятельность организована следующим образом: гостиница предоставляет номера клиентам на определенный срок. Каждый номер характеризуется вместимостью, комфортностью и ценой. Клиентами гостиницы являются различные лица, о которых администратор собирает определенную информацию (фамилия, имя, отчество и некоторый комментарий). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При поселении фиксируется дата поселения. При выезде из гостиницы для каждого места запоминается дата освобождения.

Для этого нужна общая база данных, включающая всю необходимую информацию. Программа является очень актуальной на сегодняшний день, т.к. она автоматизирует работу с базой данных. В данной дипломной работе разработана база данных для предприятия «Гостиница». Назначение разработки заключается в следующем: обеспечить удобную работу сотрудников предприятия и повысить производительность. Вся информация, касающаяся работы предприятия хранится в БД. Цель базы данных Гостиница - автоматизация процесса учета и анализа данных гостиницы.

Задачи, решаемые при использовании базы данных:

- хранение и поиск данных о сотрудниках гостиницы;
- хранение и поиск информации о клиентах гостиницы;
- хранение и поиск информации о номерах;
- хранение данных о бронировании, заселении, выезде клиентов и вычисление оплаты за проживание;
- вычисление и анализ информации о поступлении оплаты за проживание от клиентов компании;
- изменение и дополнение данных о гостинице.

1 Анализ и обзор современных тенденции развития индустрии гостеприимства

Перед тем как начать выполнение практической части дипломной работы, необходимо определить что же означает термин «предприятие гостеприимства». Итак, гостиничное предприятие – это определённый набор услуг в сфере туризма. Размещение всегда входит в пакет услуг, какими бы ни были путешествие или отдых. Так уж устроен человек, что раз в сутки он должен отдыхать, должен иметь ночлег в месте пребывания. Эти основные услуги и оказывают гостиничные предприятия разного типа и уровня обслуживания. Поэтому наличие гостиничных предприятий в том или ином регионе, туристском центре значительно влияет на возможности приема туристов. Кроме того, уровень и стандарты гостиничного обслуживания в значительной степени оказывают влияние на стандарты обслуживания туристов в целом.

Есть и другое определение гостиницы, более простое для восприятия: гостиница - это предприятие, предоставляющее людям, находящимся вне дома, комплекс услуг, важнейшими среди которых (комплексообразующими) являются в равной степени услуги размещения и питания. К основным тенденциям развития предприятий индустрии гостеприимства, оформившимся достаточно четко только в последние десятилетия можно отнести следующие:

- углубление специализации гостиничного и ресторанного предложения;
- образование международных гостиничных и ресторанных путей;
- развитие сети малых предприятий;
- внедрение в индустрию гостеприимства новых компьютерных технологий.

В последнее время наряду с традиционными отелями и ресторанами стали появляться специализированные предприятия с сокращенным набором услуг и блюд, ориентированные на обслуживание представителей определенного сегмента туристского рынка. Углубление специализации предприятий гостеприимства взаимосвязано с такой важной тенденцией, как образование международных цепей, которые играют огромную роль в разработке и продвижении высоких стандартов обслуживания. Как и многие другие тенденции, образование гостиничных цепей получило свое развитие в США.

Кроме гостиничных цепей на мировом туристском рынке активно функционируют и специфические объединения, основная цель которых – объединить лучших представителей гостиничного бизнеса. Туристские услуги, в том числе и в рамках гостиничного обслуживания, отнесены к социально-культурным услугам. Они строятся на принципах современного гостеприимства, что повышает их роль в развитии отечественного туризма, а также ставит определенные задачи в системе подготовки кадров для туристско-гостиничного сервиса.

1.1 Виды веб-приложений

Существуют различные виды веб-приложений. Каждый из них выполняет свою определенную задачу.

Системы электронной коммерции. Благодаря данной системе производители и поставщики услуг/товаров могут предлагать в сети свою продукцию покупателям, осуществлять прием и обработку заказов и т.д.

Выполняет следующие функции:

- получение подробной информации о запросах каждого потребителя;
- стремительный вывод нового продукта на рынок;
- уменьшение затрат на совершение сделки;
- сокращение пути товара к потребителю.

Корпоративный портал. Корпоративные порталы часто используют организации для оптимизации бизнес процесса. Сотрудники организации могут использовать портал для получения доступа к приложениям или нужным данным. Корпоративный портал можно ассоциировать с интранетом, как единую информационную площадку для всех сотрудников, которые могут взаимодействовать между собой. Преимуществ у корпоративных порталов очень много, пройдемся по некоторым. Сотрудники организации могут создавать различные форумы для коммуникации между собой, делится различными новостями и материалами, также создавать различные файлы и документы, которые используются при выполнении важных задач. Повышение мобильности и работоспособности сотрудников.

CRM (Customer Relationship Management). Система облегчает взаимодействие с потребителями. Система управления клиентами собирает необходимые сведения о потребителях, историю взаимодействия с ними, для повышения показателей продаж и ускорение бизнес-процесса. Сбор информации о клиентах, партнерах, поставщиков и последующий анализ данных помогают бизнесу функционировать в наиболее эффективном режиме.

Можно выделить основные принципы CRM систем: оптимизация бизнес процессов, единое хранилище данных, использование таких каналов как звонки, съезды, социальные сети, почту и т.п. Целью любой CRM системы является повышение увлеченности потребителя за счет собранных данных о клиенте.

ERP системы. Планирование ресурсов предприятия. Данное программное обеспечение помогает максимально эффективно использовать ресурсы предприятия. ERP система помогает синхронизировать и управлять процессами в предприятии, а также улучшает взаимодействие между отделами компаний и экономит трудовые ресурсы.

ITRP (IT Resources Planning) — класс Web-приложений, предназначен для поддержки управления корпоративными ИТ-ресурсами и сервисами.

OSS (Operation Support Systems) — вид Web-приложений, предназначен для обеспечения работы операторов распределенных вычислительных сетей. OSS обеспечивает управление сетью, производительность, ликвидацию сбоев в работе, создание и учет сервисов, планирование сетевых ресурсов, мониторинг процессов, контроль за безопасностью, качество услуг и уровень обслуживания

клиентов, сбором статистических данных. Разновидностью OSS является система поддержки бизнеса — BSS (Business Support Systems). К ним относятся биллинговые системы, системы управления взаимоотношениями с клиентами, управления сетями, заказами, качеством услуг.

Специализированные веб-сайты. Эти сайты имеют свою тематику и целевую аудиторию. К специализированным веб-сайтам можно отнести: блог, интернет-магазин, социальную сеть, форум, игровые сайты и т.п. Социальные сети используются для общения с людьми в интернете. Также каждый пользователь может создавать профиль с личной информацией о себе. Выкладывать различные фотографии, видео и новости. Социальные сети используются как для бизнеса, так и для развлечений.

1.2 Архитектура клиент-сервер

В настоящее время компьютерные транзакции, в которых сервер выполняет запрос, отправленный клиентом, очень распространены, и модель клиент-сервер стала одной из главных идей сетевых вычислений. Клиент устанавливает соединение с сервером через локальную (LAN) или глобальную сеть (WAN). Как только сервер выполнил запрос клиента, соединение разрывается. Поскольку несколько клиентских программ совместно используют службы одной и той же серверной программы, специальный сервер, называемый daemon, может быть активирован только для ожидания клиентских запросов. Важным преимуществом модели клиент-сервер является то, что ее централизованная архитектура помогает упростить защиту данных с помощью средств управления доступом. Кроме того, не имеет значения, построены ли клиенты и сервер в одной операционной системе, поскольку данные передаются по протоколам клиент-сервер, которые не зависят от платформы.

Недостатком модели клиент-сервер является то, что, если слишком много клиентов одновременно запрашивают данные с сервера, они могут быть перегружены. В дополнение к перегрузке сети, слишком много запросов может привести к отказу в обслуживании.

Приложение организуется по так называемому паттерну MVC. В Django этот паттерн называют MVT, но сути это не меняет. Что из себя представляет MVC: Модель (Model), Представление (View), Контроллер (Controller). Модель отвечает за работу с данными. Она взаимодействует с базой данных, проверяет данные на корректность, создает расчетные данные в случае необходимости. Представление отвечает за то, как эти данные будут выглядеть, придает им удобочитаемую форму. Контроллер является прослойкой между моделью, представлением и посетителем сайта. Когда посетитель сайта запрашивает страницу, этот запрос передается контроллеру. Он в свою очередь собирает все модели, которые могу пригодиться, и просит их предоставить данные. Потом контроллер выбирает представление, которое лучше всего отвечает запросу пользователя и передает ему данные. Сформированная страница (представление)

передается как ответ на запрос. В Django начинается небольшая путаница, поскольку в его терминах этот паттерн называется Model-View-Template (Модель-Представление-Шаблон). Однако View в Django скорее выполняет функцию контроллера, а Template - представления.

1.3 Анализ существующих технологий создания веб-приложений

При разработке программного обеспечения моего веб-приложения, основным принципом было создать программу с удобным и понятным интерфейсом, которая бы облегчило онлайн бронирование номеров, была легкой и понятной для обслуживающего персонала в частности для администратора. Для разработки программного обеспечения идеально подходит инструмент для создания программ самого разнообразного назначения этот инструмент язык программирования Python. С помощью которого можно решать задачи различного типа.

Язык Python невероятно эффективен: ваши программы делают больше, чем многие другие языки, в меньшем объеме кода. Синтаксис Python также позволяет писать «чистый» код. Ваш код будет легко читаться, у вас будет меньше проблем с отладкой и расширением программ по сравнению с другими языками. Python используется для разных целей: для создания игр, построения веб-приложений, решений бизнес-задач и разработки внутренних инструментов для всевозможных интересных проектов. Python также широко применяется в научной области для теоретических исследований и решения прикладных задач. Python содержит большой набор инструментов для создания веб-приложений.

Django фреймворк для веб-приложений на языке Python. Один из основных принципов фреймворка — DRY (don't repeat yourself). Веб-системы на Django строятся из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из заметных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Также, в отличие от многих других фреймворков, обработчики URL в Django конфигурируются явно (при помощи регулярных выражений), а не автоматически задаются из структуры контроллеров.Django проектировался для работы под управлением Apache (с модулем mod_python) и с использованием PostgreSQL в качестве базы данных. В настоящее время, помимо PostgreSQL, Django может работать с другими СУБД: MySQL (MariaDB), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle. Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (Templates). Из-за этого

уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).

Первоначально разработка Django велась для обеспечения более удобной работы с новостными ресурсами, что достаточно сильно отразилось на архитектуре: фреймворк предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав). Веб-фреймворк Django используется в таких крупных и известных сайтах, как Instagram, Disqus, Mozilla, The Washington Times, Pinterest, lamoda и др

Скриптовые языки. Большинство современных Web-браузеров способно интерпретировать код на скриптовых языках, таких как VBScript и JavaScript. Код на этих языках внедряется в Web-страницу и интерпретируется браузером. Типичный пример применения скриптовых языков — проверка корректности данных, вводимых пользователем в соответствующие поля HTML-формы, непосредственно в процессе ввода или после него, без обращения к Web-серверу. Подобные примеры применения скриптовых языков можно обнаружить при заполнении некоторых анкет и получении сообщений о том, что не заполнены обязательные поля (справедливости ради отметим, что далеко не все анкеты реализованы подобным образом).

ASP, JSP, PHP. Очередной шаг в развитии технологий создания Интернет-приложений — появление средств, позволяющих отделить задачи Web-дизайна от задач, связанных с реализацией функциональности приложений. Первой из таких технологий стала Active Server Pages (ASP), построенная на основе ISAPI-фильтра. Основная идея ASP заключается в создании Web-страниц с внедренными в них фрагментами кода на скриптовых языках. Однако, в отличие от рассмотренных выше средств применения скриптовых языков для расширения функциональности браузеров, указанные фрагменты кода интерпретируются не браузером, а сервером (точнее, предназначенной для этого ISAPI-библиотекой), и результат выполнения этих фрагментов кода замещает сам фрагмент кода в той версии страницы, которая передается в пользовательский браузер. Вскоре после ASP появились и другие технологии, реализующие идею размещения внутри Web-страницы кода, выполняемого Web-сервером. Наиболее известной из них сегодня является технология JSP (Java Server Pages), основная идея которой — однократная компиляция Java-кода (сервлета) при первом обращении к нему, выполнение методов этого сервлета и помещение результатов выполнения этих методов в набор данных,

отправляемых в браузер. Еще одной популярной технологией подобного типа является PHP (Personal Home Pages), которая использует CGI-приложения, интерпретирующие внедренный в HTML-страницу код на скриптовом языке. Новейшей версией технологии Active Server Pages является ASP.NET, ключевая в архитектуре Microsoft.NET Framework. Основное отличие этой технологии от ASP с точки зрения архитектуры приложений заключается в том, что код, присутствующий на Web-странице, не интерпретируется, а компилируется и кэшируется, что, естественно, способствует повышению производительности приложений. С помощью ASP.NET можно создавать Web-приложения и Web-сервисы, которые не только позволяют реализовать динамическую генерацию HTML-страниц, но и интегрируются с серверными компонентами и могут использоваться для решения широкого круга бизнес-задач, возникающих перед разработчиками современных Web-приложений.

Основная проблема всех CGI-приложений заключается в том, что при каждом клиентском запросе сервер загружает это приложение в отдельное адресное пространство, а затем инициирует его выполнение и выгрузку. Эта особенность ограничивает производительность приложений и возможность одновременной обработки большого количества клиентских запросов. ISAPI и Apache DSO. Проблему ограниченной производительности Web-приложений, которые выполняются в отдельном адресном пространстве, можно решить, создав приложение в виде библиотеки, загружающейся в адресное пространство Web-сервера и при необходимости остающейся там для обработки последующих запросов от других клиентов; естественно, в этом случае Web-сервер должен поддерживать загрузку таких библиотек. Подобные приложения для Microsoft Internet Information Servise носят название ISAPI (Internet Server Application Program Interface), а для весьма популярного Web-сервера Apache такие библиотеки называются Apache DSO (Dynamic Shared Objects). Отметим, однако, что при создании как CGI-, так и ISAPI-приложений было довольно сложно отделить задачи Web-дизайна от задач, связанных с реализацией функциональности и логики приложений, подобные приложения генерируют Web-страницы целиком, поэтому все данные, связанные с дизайном этих страниц, должны в общем случае содержаться внутри исполняемого файла.

Приложения Macromedia Flash являются сегодня наиболее популярным расширением функциональности Web-браузеров — с их помощью многие Web-дизайнеры придают своим сайтам интерактивность и оригинальность. Модель безопасности приложений Flash основана на том, что Macromedia Flash Player, как и виртуальная Java-машина, выполняет приложения в ограниченном адресном пространстве, при этом выполняемые приложения не имеют доступа к файловой системе (кроме одного конкретного каталога, используемого Macromedia Flash Player для служебных целей) и другим ресурсам компьютера пользователя; исключение делается для микрофонов и видеокамер, однако пользователь должен дать разрешение на передачу данных, полученных с этих устройств. Доступ к сетевым ресурсам ограничивается доменом, с которого было

получено приложение. Отметим, что приложения Flash также могут управляться с помощью кода JavaScript, присутствующего на той же странице. Сам Macromedia Flash Player для Microsoft Internet Explorer является элементом управления ActiveX и использует возможности элементов управления ActiveX для доступа к свойствам приложений Flash из скриптовых языков.

Web-сервисы. Говоря о серверных Web-технологиях, нельзя обойти вниманием такую важную, как Web-сервисы XML. На Web-сервисы XML в настоящее время нередко возлагается решение многих задач, связанных с интеграцией приложений, в том числе созданных на разных платформах. Создавать Web-сервисы можно и в виде исполняемых файлов, и в виде библиотек, и в виде интерпретируемого кода; существуют также средства представления бизнес объектов в виде Web-сервисов. Методы Web-сервисов можно вызывать из обычных приложений, Web-приложений и других Web-сервисов, и, за редким исключением, конечные пользователи непосредственно с Web-сервисами дела не имеют. Тем не менее, в последнее время отмечается массовое появление приложений, использующих Web-сервисы, в том числе и приложений, предназначенных для конечных пользователей. В языке XML все имена должны начинаться с буквы, символа подчёркивания (_) или двоеточия (:) и продолжаться только допустимыми для имён символами, а именно они могут содержать только буквы, входящие в секцию букв кодировки Unicode, арабские цифры, дефисы, знаки подчёркивания, точки и двоеточия. Однако имена не могут начинаться со строки xml в любом регистре. Имена, начинающиеся с этих символов, зарезервированы для использования консорциумом W3C. Нужно помнить, что так как буквы не ограничены исключительно символами ASCII, то в именах можно использовать слова из родного языка.

JPA – это технология, обеспечивающая объектно-реляционное отображение простых JAVA объектов и предоставляющая API для сохранения, получения и управления такими объектами. JPA – это спецификация (документ, утвержденный как стандарт, описывающий все аспекты технологии), часть EJB3 спецификации. Сам JPA не умеет ни сохранять, ни управлять объектами.

Java-апплеты. Практически все современные браузеры способны отображать и выполнять Java-апплеты — специальные Java-приложения, которые пользователь получает в составе Web-страницы. Эти приложения нередко включаются в состав Web-страниц с целью добавления функциональности, которую сложно или невозможно реализовать с помощью скриптовых языков. Апплеты могут выполняться на всех платформах, для которых доступна виртуальная Java-машина.

1.4 Постановка задач дипломной работы

Цель дипломной работы: создание веб-приложения для автоматизации и упрощения онлайн бронирования номеров клиентами. Требования к функциональным характеристикам. Приложение должно обеспечивать возможность выполнения следующих функций:

- приложение должно обладать удобным и понятным интерфейсом для пользователей;
- в приложении должна быть предусмотрена возможность онлайн бронирования и прием заявок от клиентов;
- приложение должно предусматривать контроль вводимой информации;
- приложение должно обладать возможностью удаления профиля;
- программный код приложения должен комментироваться на этапе разработки для будущих изменений или дополнений;
- приложение должно предоставлять список услуг для пользователей;
- в приложении должен быть раздел отзывов для пользователей;
- приложение должно иметь окна регистрации и авторизации пользователя;
- в приложении должна присутствовать административная часть, где происходит контроль за всеми происходящими действиями в приложения. Окно приложения показано на рисунке 1.1

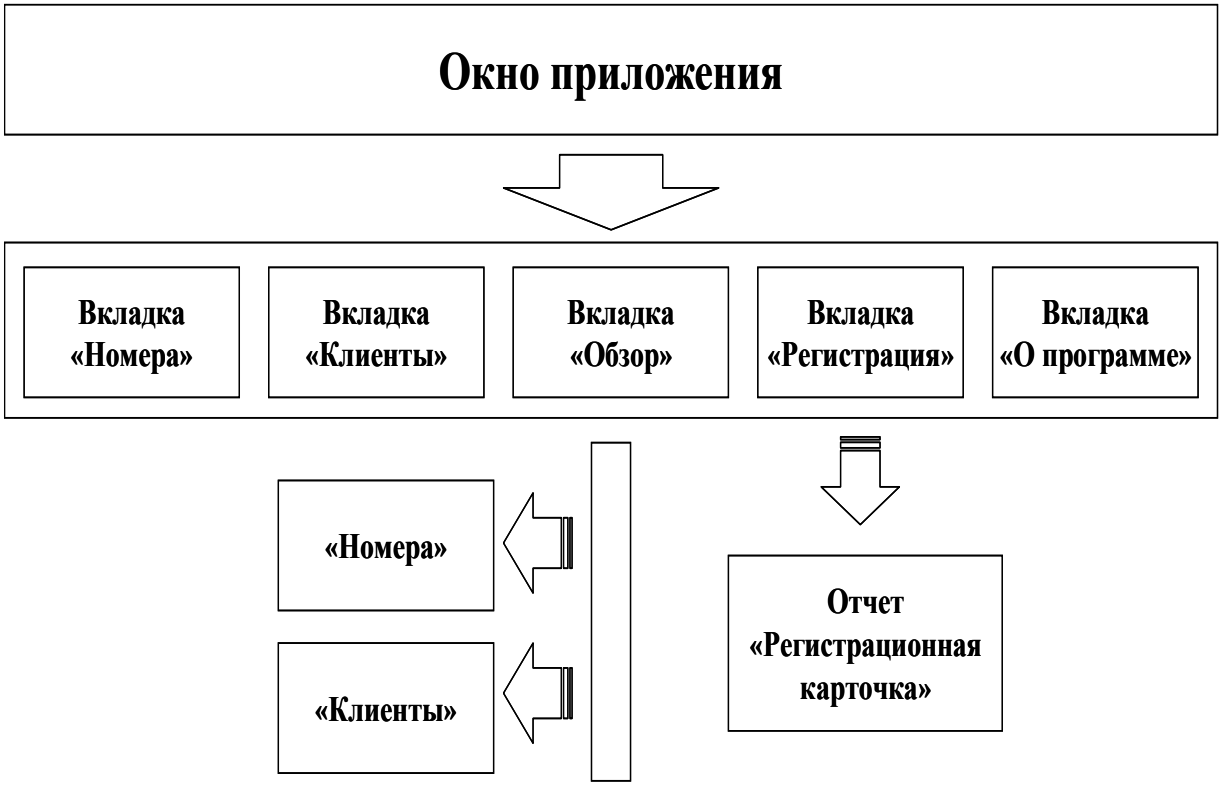

Рисунок 1.1- Окно приложения

2 Модели и службы функционирования системы

2.1 Функциональная модель системы

Опишем основные функциональные возможности веб-приложения. При всем многообразии форм представления гостиниц в Интернете схема заказа номера остается более или менее общей:

- поиск нужной гостиницы по ряду критериев (цена или расстояние до аэропорта и др.);
- подробное изучение правил работы с выбранной гостиницей (сроки бронирования и отказа от бронирования, комиссионные агентам и др.);
- заполнение формы бронирования (сроки проживания, имена клиентов и т.д.);
- получение либо подтверждения о приеме бронирования в обработку, либо подтверждения самого бронирования.

Принципиально отличаются бронирования, сделанные на трех типах серверов: на сервере гостиницы или гостиничной цепочки, на сервере туроператора и на сервере центра резервирования. В гостинице должны быть установлены автоматизированные рабочие места для персонала, которые обеспечивают автоматизацию основных функциональных процессов.

Все автоматизированные рабочие места объединяются в локальную компьютерную сеть, которая организует взаимодействие между отделами, уменьшая время для обеспечения коммуникации, и позволяет эффективно наладить работу всей гостиницы.

В системе онлайн обслуживания можно:

- бронировать места и номера;
- следить за загрузкой номеров;
- регистрировать клиентов;
- вести картотеку гостей;
- производить расчеты;
- составлять финансовые отчеты.

Все эти функций улучшают работу и обслуживание клиентов.

2.2 Обобщенный алгоритм функционирования системы

Информационная система состоит из двух составляющих:

- клиентская часть;
- серверная часть.

Клиентская часть реализует пользовательский интерфейс, создает запросы к серверу и обрабатывает ответы от него. Серверная часть принимает запрос от клиента, производит обработку, после, формирует и отправляет страницу клиенту с использованием протокола HTTP.

Приложение включает в себя следующие подсистемы:

- аутентификация сотрудника – проверка подлинности сотрудника по авторизованной учетной записи, не позволяющая осуществить несанкционированный доступ;

- авторизация клиентов в систем – предоставление прав доступа и выявления ролей у пользователя;
- новый сотрудник – процедура создания новых сотрудников системы, с распределением ролей;
- онлайн бронирование – процедура бронирования номеров гостиницы;
- приема заявок – процедура принятия заявок от клиентов;
- наличие номеров и их разновидности – предоставление списка номеров и их разновидностей;
- дата заезда и дата выезда – информация для пользователя и администратора;
- список персонала – предоставление списка всех сотрудников пользователю;
- список услуг – предоставления списка всех услуг пользователю;
- раздел для отзывов – модуль для оставления отзывов от пользователей;
- панель управления гостиницей для администратора.
- регистрация клиентов – для добавления новых пользователей.

В работе имеются алгоритмы, реализующие функционирование системы, учитывая степень развития современных информационных технологий, были разработаны и предложены новые алгоритмы, которые представлены ниже.

Рассмотрим работу алгоритмов приложения более подробно.

При аутентификации сотрудник, посредством ввода необходимых данных (логин, пароль) формирует и отправляет запрос на сервер. Серверная часть обрабатывает запрос, и с помощью специальных методов производит сверку введенных логина и пароля с данными, хранящимися в базе. Они должны совпадать. В случае успешной проверки, формируется токен пользователя. Токен (JWT – JSON Web Token) – это набор символов, полученный шифрованием некой информации о пользователе, прошедшем аутентификацию в приложении. Любой, кто прошел аутентификацию получает токен и этот же токен запоминается еще и системой выдавшей его, чтобы при последующих запросах от этого пользователя сверять переданный им токен с тем, что система запомнила в момент успешной аутентификации. Если токен не совпадает, либо истек срок его годности (срок годности выбирается разработчиком на свое усмотрение), то запрос считается не прошедшим авторизацию и отклоняется системой.

При добавлении нового пользователя необходимо перейти на страницу добавления пользователя. Для этого нужно обладать соответствующими привилегиями, чтобы система разрешила доступ к добавлению. Если роль пользователя не соответствует, то запрос отклоняется системой. Далее, необходимо заполнить поля, следующими данными клиента: ФИО, ИИН, номер удостоверения личности, как показано на рисунке 1.1

Рисунок 1.1- Схема алгоритм функционирования системы

2.3 Обоснование выбора СУБД

База данных (БД) - это система для хранения и обработки взаимосвязанных данных.

СУБД позволяет пользователям создавать, считывать, обновлять и удалять данные из баз данных. СУБД выступает в качестве интерфейса между базой данных и конечным пользователем или приложением, обеспечивая согласованную организацию данных и легкий доступ.

SQL (язык структурированных запросов) - это стандартизированный язык программирования для управления реляционными базами данных и выполнения различных задач обработки данных. Использование SQL включает изменение таблицы базы данных и структуры индекса; добавление, обновление и удаление строк данных; и извлечение подмножеств информации из базы данных для приложений обработки транзакций и аналитики. Запросы и другие операции SQL принимают форму команд, написанных как операторы. Обычно используемые операторы SQL включают выбор, добавление, вставку, обновление, удаление, создание, изменение и усечение. в SQL существует специальный набор общих команд — таких, как SELECT, INSERT, UPDATE и DELETE.

Классификация БД происходит по следующим критериям:

- сфера использования;
- инфраструктура;
- характеристики обработки;
- количество полученных данных;
- уровни классифицирования хранящихся данных;
- качества исследуемых объектов.

Работать с БД без СУБД, которые могут преобразовать большие объемы данных в поддающуюся влиянию информацию, было бы невозможным. Чтобы обработать данные больших объемов, сотруднику понадобится много времени, помимо этого могут возникнуть проблемы с внесением данных, так как сотрудник может забыть какую-то важную информацию.Правильная организация базы данных обеспечивает более быстрые и гибкие возможности выборки данных. Она существенно упрощает реализацию средств поиска и сортировки, а проблемы прав доступа к информации решаются при помощи средств контроля за привилегиями, присутствующими во многих системах управления базами данных (СУБД). Кроме того, упрощаются процессы репликации и архивации данных [2].

Язык SQL заложен в основу практически всех существующих СУБД. Чтобы перейти к рассмотрению примеров работы с базами данных, необходимо хотя бы в общих чертах представлять, как работает SQL.

Для создания базы данных используем СУБД SQLite. SQLite компактная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние. Является чисто реляционной базой данных. Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер. Т.е. движок SQLite не является отдельно работающим процессом, с которым

взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID [1]-функции достигаются в том числе за счёт создания файла журнала. Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени. Можно, также, ввести таймаут операций. Тогда подключение, столкнувшись с занятостью БД, будет ждать N секунду прежде, чем отвалиться с ошибкой SQLITE_BUSY.

Также с версии 3.7.0 присутствует режим WAL [2], с помощью которого можно испольовать одну и ту же базу несколькими приложениями, как на чтение, так и на запись.В комплекте поставки идёт также функциональная клиентская часть в виде исполняемого файла sqlite3, с помощью которого демонстрируется реализация функций основной библиотеки. Клиентская часть работает из командной строки, позволяет обращаться к файлу БД на основе типовых функций ОС.Благодаря архитектуре движка возможно использовать SQLite как на встраиваемых системах, так и на выделенных машинах с гигабайтными массивами данных.Формат файла базы данных является кросс-платформеным, что позволяет без проблем использовать одну и ту же базу на нескольких операционных системах. Также присутствует возможность хранения базы в памяти, без её записи на диск. Этот вариант используется по умолчанию для консольной утилиты sqlite3, если не указано имя файла.

Поддержка стандарта SQL. Как известно, в своем развитии SQL устремился в разные стороны. Крупные производители начали впихивать всякие расширения. И хотя принимаются всякие стандарты, в реальной жизни все крупные БД не поддерживают стандартов полностью. Но зато имеют что-то свое. Так вот, SQLite старается жить по принципу «минимальный, но полный набор». Она не поддерживает сложные штуки, но во многом соответствует SQL 92. И вводит некие свои особенности, которые очень удобны, но — не стандартны.

Не поддерживаются следующие возможности:

- RIGHT и FULL OUTER JOIN, реализован только LEFT OUTER JOIN;
- частично реализован ALTER TABLE, доступны только RENAME TABLE и ADD COLUMN;

- частичная поддержка тригеров, доступны только FOR EACH ROW тригеры;
- запись во VIEWS доступны только на чтение, частично обходится через тригеры;
- в силу реализации базы данных, как единственного файла и отхода от концепции «клиент-сервер», не используются возможности GRANT и REVOKE.

По умолчанию отключены внешние ключи. Это сделано для обратной совместимости.

Типизация. SQLite использует динамическое типизирование данных. Это значит, что тип столбца не определяет тип хранимого значения в этом поле записи. Т.е. в любой столбец можно занести любое значение.Тип столбца определяет как сравнивать значения (нужно же их привести к единому типу при сравнении, скажем, внутри индекса). Но не обязывает заносить значения именно такого типа в столбец. Допустим, мы объявили столбец как «A INTEGER». SQLite позволяет занести в этот столбец значения любого типа (999, «abc», «123», 678.525). Если вставляемое значение — не целое, то SQLite пытается привести его к целому. Т.е. строка «123» превратится в целое 123, а остальные значения запишутся «как есть». Возможные типы полей: NULL, INTEGER, REAL, TEXT, BLOB.

Надёжность. Ситуация с покрытиями тестами исходного кода SQLite в некотором роде является легендой. Это связно с тем, что тестами описано практически все возможные (и невозможные) ситуации. Кода, описывающего тесты, намного больше, чем кода реализующего SQLite. Естественно, что покрытие кода тестами 100%. Сам же подход к тестированию использующийся разработчиками SQLite достоин подражания. И может быть взят как идеал, к которому следуют стремиться.

В качестве практического занятия для закрепления понимания реляционной модели и иллюстрации её ограниченности, мы рассмотрим пример создания структуры каталога товаров. Для того чтобы запустить SQL-код необходимо выполнить следующую команду: sqlite3 ~/example.sqlite. В качестве практического занятия для закрепления понимания реляционной модели и иллюстрации её ограниченности, мы рассмотрим пример создания структуры каталога товаров.Для того чтобы запустить SQL-код необходимо выполнить следующую команду: sqlite3 ~/example.sqlite. Далее откроется интерактивный SQLite-shell, куда можно вводить команды. Крупные производители начали впихивать всякие расширения. И хотя принимаются всякие стандарты, в реальной жизни все крупные БД не поддерживают стандартов полностью. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID [1]-функции достигаются в том числе за счёт создания файла журнала.

2.4 Разработка структуры базы данных

В составе БД имеется три основные таблицы:

- 1 таблица – содержит в себе данные, характеризующие каждый гостиничный номер, а именно: Номер, количество комнат, категория номера и стоимость;
- 2 таблица – содержит данные о клиентах, с которым гостиница предоставляла услуги, а именно – ФИО клиента, домашний адрес и паспортные данные;
- 3 таблица – содержит данные о регистрации: уникальный код гостиничного номера, уникальный код клиента, дата прибытия, дата убытия и стоимость проживания.

Это то место, где будут размещаться таблицы и индексы. Позднее они будут скопированы в каталог, специально предназначенный для приложения.

Из меню File (Файл) в DBD нужно выбрать New/Templates новый html документи выбрать dBase для Windows из предложенного набора типов файлов. Открывшийся после этого диалог CreateTemplates (Создать таблицу) помогает определить таблицу с помощью специфики полей. Пользовательский интерфейс с БД выполнен на русском языке. Скорость работы БД, как и других программных продуктов, в значительной степени зависит от используемого оборудования. Как показала практика, использование компьютера на базе процессора Pentium 100 вполне достаточно для работы в реальном времени.

Концептуальная модель

При анализе предметной области был выявлен следующий перечень сущностей:

- гостиничный номер;
- клиент;
- регистрация клиентов.

Таблица 1 – Физическая таблица модели данных

Имя в программе	Тип	Длина	Назначение
NOMER	+ (Autoincrement)		Номер записи
N_KOMN	hms		Гостиничный номер
KOL_KOMN	hms		Количество комнат
KLASS	hms	15	Категория номера
CENA	hms		Стоимость

Таблица 2 – Клиенты (Klients.db)

Имя в программе	Тип	Длина	Назначение
NOMER	+ (Autoincrement)		Номер записи
FIO	main	50	ФИО клиента
ADRES	main	100	Адрес клиента
PASSPORT	main	50	Паспортные данные

Таблица 3– Регистрация (registration)

Имя в программе	Тип	Длина	Назначение
Logged_out	+ (Autoincrement)		Номер записи
login	system		Код клиента
Password_reset	system		Код гост. номера
Password_reset_confirm	system		Дата прибытия
Password_reset_form	system		Дата убытия

Схема работы программы как показано на рисунке 1.3

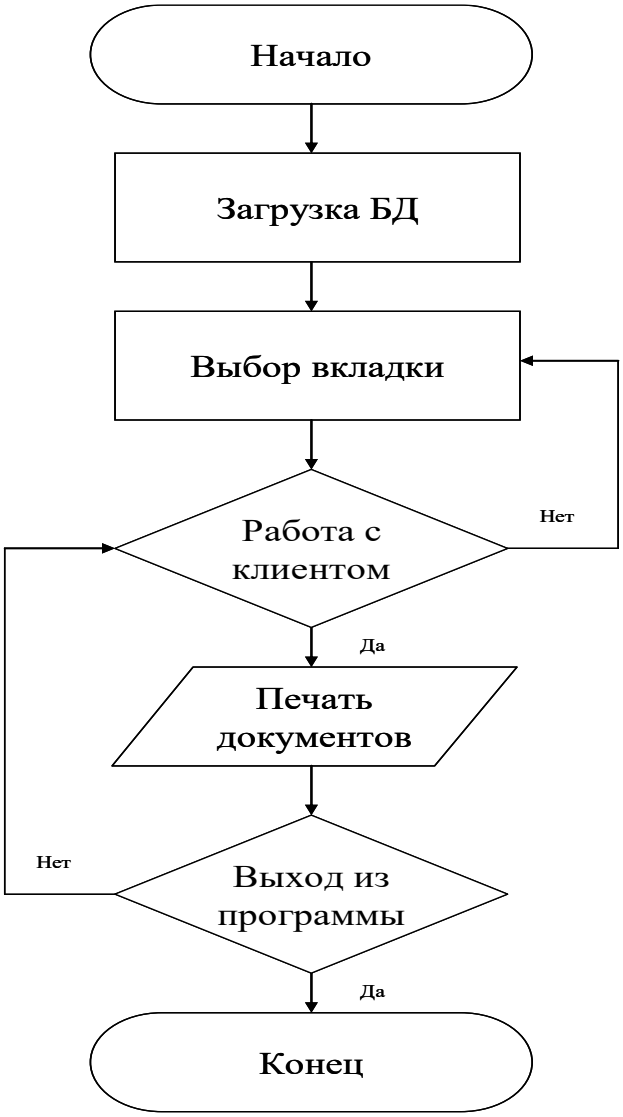

Рисунок 1.3 – Схема работы программы

3 Информационное и программное обеспечение

3.1 Структура программного обеспечения

Программное обеспечение ИС «Гостиница» состоит из:

- общего (системного) обеспечения, включающего ОС Windows 10;
- функционального (прикладного) обеспечения, включающего набор программных модулей, обеспечивающих автоматизированное выполнение основных операций, производимых валютным кассиром в процессе работы;
- интегрированной среды программирования Python, обеспечивающей с помощью своих средств интерфейс между пользовательскими программами и ОС, с одной стороны, и работу с БД, с другой.

В полной спецификации описываются цели проекта, его функциональность, а также внешний вид и интерфейс пользователя. Как и любой хороший проект или бизнес-план, спецификация должна сосредоточиться на самых важных аспектах и обеспечивать планомерную разработку проекта. Здесь мы не будем писать полную спецификацию, а сформулируем несколько четких целей, которые будут задавать направление процесса разработки. Вот как выглядит спецификация: Важной задачей при разработке программного обеспечения является создание и поддержка в среде Python БД и связка htmlстраниц между собой что бы у нас получилось единая система, которая является информационной основой для работы автоматизированной системы в целом. Специфика работы обменного пункта состоит в том, что надо при настройке системы на текущий день, а зачастую и в течение рабочего дня, регулярно обновлять входные данные. Поэтому простота доступа к таблицам БД и широкие возможности их редактирования являются непременным условием при разработке программного обеспечения

Во время изучения нового материала бывает полезно вести журнал того, что вы узнали, — записи пригодятся для контроля и возвращения к необходимой информации. Хорошее приложение повышает эффективность этого процесса. В начале работы над проектом необходимо описать проект в спецификации. Затем вы создадите виртуальную среду для построения проекта.

Для работы с Django необходимо сначала создать виртуальную среду для работы. Виртуальная среда представляет собой подраздел системы, в котором вы можете устанавливать пакеты в изоляции от всех остальных пакетов Python. Отделение библиотек одного проекта от других проектов принесет пользу при развертывании на сервере. Создадим для проекта новый каталог, и перейдем в этот каталог в терминальном режиме и создадим виртуальную среду. Мы работаем с Python 3, и создаем виртуальную среду следующей командой: python -m venvenv Команда запускает модуль venv и использует его для создания виртуальной среды с именем env. Первоначально разработка Django велась для обеспечения более удобной работы с новостными ресурсами, что достаточно сильно отразилось на архитектуре.

3.2 Системное программное обеспечение

Так как Django хранит бульшую часть информации, относящейся к проекту, в базе данных, необходимо создать базу данных, с которой Django сможет работать. Чтобы создать базу данных для нашего проекта, введем следующую команду (все еще не покидая активной среды): python manage.py migrateOperationstoperform: Synchronizeunmigratedapps: messages, staticfilesApplyallmigrations: contenttypes, sessions, auth, admin ... Applying sessions.0001_initial... OK ls db.sqlite3 learning_logll_env manage.py Каждое изменение базы данных называется миграцией. Первое выполнение команды migrate приказывает Django проверить, что база данных соответствует текущему состоянию проекта. Когда мы впервые выполняем эту команду в новом проекте с использованием SQLite (вскоре мы расскажем о SQLite более подробно), Django создает новую базу данных за нас. В точке Django сообщает о создании таблиц базы данных, необходимых для хранения информации, используемой в проекте. Знакомство с Django (Synchronizeunmigratedapps), а затем проверяет, что структура базы данных соответствует текущему коду (Applyallmigrations). Выполнение команды ls показывает, что Django создает другой файл с именем db.sqlite3. SQLite — база данных, работающая с одним файлом; она идеально подходит для написания простых приложений, потому что вам не нужно особенно следить за управлением базой данных.

В большинстве случаев переходы между этими состояниями выполняются автоматически, однако следует иметь ввиду , сто существует много событий, относящихся к переходам между состояниями.

Убедимся в том, что проект был создан правильно. Введитекомандуrunserver: (env) python manage.py runserver Performing system checks... System check identified no issues (0 silenced). Django, using settings Starting development server at http://127.0.0.1:8000/ Quit the server with CONTROL-C. Django запускает сервер, чтобы вы могли просмотреть проект в своей системе и проверить, как он работает. Когда вы запрашиваете страницу, вводя URL в браузере, сервер Django отвечает на запрос; для этого он строит соответствующую страницу и отправляет страницу браузеру. В точке Django проверяет правильность созданного проекта; в точке выводится версия Django и имя используемого файла настроек; в точке возвращается URL-адрес, по которому доступен проект. URL http://127.0.0.1:8000/ означает, что проект ведет прослушивание запросов на порте 8000 локального хоста (localhost), то есть вашего компьютера. Термином «локальный хост» обозначается сервер, который обрабатывает только запросы вашей системы; он не позволяет никому другому просмотреть разрабатываемые страницы. Теперь откройте браузер и введите URL http://localhost:8000/ — или http://127.0.0.1:8000/, если первый адрес не работает. Пока не завершайте работу сервера (но, когда вы захотите прервать ее, это можно сделать нажатием клавиш Ctrl+C). Структура программы, использующая объектное программирование, существенно отличается от

традиционной структуры программ с жестким, заранее заданным алгоритмом. Здесь программа выглядит как совокупность, в некотором смысле, самостоятельных, обособленных блоков, выполняющих те или иные операции, а связь между ними определяется результатами предыдущих этапов и взаимодействием программы через внешние устройства с пользователем. После выполнения очередного блока программа приостанавливается и дожидается сообщения от оператора, которое через ОС Windows 10 передаётся программе.

Структурно программа формируется из модулей, выполняющих отдельные операции, входящие в состав функционального набора ИС.

3.3 Инструментальное программное обеспечение

Для программной реализации был выбран высокоуровневый язык программирования python.

Python как многоцелевой язык программирования, практически не ограничен: он может использоваться для реализации чего угодно, от веб-сайтов и игровых программ до управления роботами и космическими кораблями.

Встроенные в Python интерфейсы доступа к службам операционных систем делают его идеальным инструментом для создания переносимых программ и утилит системного администрирования. Программы на языке Python могут отыскивать файлы и каталоги, запускать другие программы, производить параллельные вычисления с использованием нескольких процессоров и потоков, и делать многое другое. Также поддерживает все типичные инструменты ОС: файлы, сокеты, каналы, процессы, многопоточную модель исполнения, поиск по шаблону с использование рег. выражений, аргументы ком. строки и много другое.

Помимо всего этого для Python существуют полноценные пакеты веб-разработки, такие как Django, Zope, TurboGears, Pylons и WebWare, предназначенные для быстрого создания полнофункциональных высококачественных веб-сайтов.Джанго – это высокоуровневый питоновский Web-фреймворк, который реализован на основе архитектуры MVC. Джанго имеет прозрачный дизайн, дает возможность для оперативной разработки Web-приложений, позволяет разрабатывать динамические Web-сайты.

Отличительные особенности джанго:

- любой запрос обрабатывается программно и перенаправляется на свой адрес(url);
- разделение контента и представления с помощью шаблонов;
- абстрагирование от низкого уровня баз данных;

Джанго-приложение состоит из четырех основных компонентов. Модель данных являются сердцевиной любого современного Web-приложения. Модель – важнейшая часть приложения, которое постоянно обращается к данным при любом запросе из любой сессии. Любая модель является стандартным питоновским классом. Объектно-ориентированный маппер (ORM) обеспечивает

таким классам доступ непосредственно к базам данных. Если бы не было ORM, программисту пришлось бы писать запросы непосредственно на SQL. Модель обеспечивает облегченный механизм доступа к слою данных, инкапсулирует бизнес-логику. Модель не зависит от конкретного приложения. Данными можно манипулировать даже из командной строки, не используя при этом Web-сервер.

Представление (view). Вьюхи в джанге выполняют разнообразные функции, в том числе контролируют запросы пользователя, выдают контекст в зависимости от его роли. View – это обычная функция, которая вызывается в ответ на запрос какого-то адреса (url) и возвращает контекст.

Шаблоны: они являются формой представления данных. Шаблоны имеют свой собственный простой метаязык и являются одним из основных средств вывода на экран.

URL всего лишь механизм внешнего доступа к представлениям (view). Встроенные в урлы регулярные выражения делают механизм достаточно гибким. При этом одно представление может быть сконфигурировано к нескольким урлам, предоставляя доступ различным приложениям. Здесь поддерживается философия закладок: урлы становятся как бы самодостаточными и начинают жить независимо от представления.

3.4 Прикладное обеспечение

На прикладном уровне такие архитектурные решения поддержаны конкретными инструментами разработки ПО. При этом последние представлены в виде модульных программных каркасов — фреймворков, объединяющих множество отдельных компонентов для решения прикладных задач. Расширение типовых функциональных возможностей интегрированной среды разработки возможно за счет подключения дополнительных библиотек и пакетов. Наиболее наглядными примерами подобных инструментальных средств сегодня являются фреймворки для разработки веб-приложений: Ruby on Rails, Pylons, Phalcon, Django.

Использование таких программных каркасов позволяет ускорить процесс разработки за счет автоматизированного формирования базы данных на основе моделей, пользовательского интерфейса и функциональных компонентов. Дополнительным преимуществом является наличие у современных языков программирования и сред разработки инструментов для тестирования ПО. Например, модули doctext и unittest в стандартной библиотеке языка Python, TextTest — кроссплатформенный фреймворк для тестирования графического интерфейса пользователя, Selenium WebDriver — для автоматического тестирования веб-интефейсов и другие подобные системы. Таким образом, лучшие практики проектирования и программирования унифицируются и в дальнейшем используются при разработке новых продуктов в качестве так называемых «кирпичиков».

3.5 Функциональное назначение

При всем многообразии форм представления гостиниц в Интернете схема заказа номера остается более или менее общей:

- поиск нужной гостиницы по ряду критериев (цена или расстояние до аэропорта и др.);
- подробное изучение правил работы с выбранной гостиницей (сроки бронирования и отказа от бронирования, комиссионные агентам и др.);
- заполнение формы бронирования (сроки проживания, имена клиентов и т.д.);
- получение либо подтверждения о приеме бронирования в обработку, либо подтверждения самого бронирования.

Принципиально отличаются бронирования, сделанные на трех типах серверов: на сервере гостиницы или гостиничной цепочки, на сервере туроператора и на сервере центра резервирования.

В гостинице должны быть установлены автоматизированные рабочие места для персонала, которые обеспечивают автоматизацию основных функциональных процессов.

Все автоматизированные рабочие места объединяются в локальную компьютерную сеть, которая организует взаимодействие между отделами, уменьшая время для обеспечения коммуникации, и позволяет эффективно наладить работу всей гостиницы.

В системе онлайн обслуживания можно:

- бронировать места и номера;
- следить за загрузкой номеров;
- регистрировать клиентов;
- вести картотеку гостей;
- производить расчеты;
- составлять финансовые отчеты.

Все эти функций улучшают работу и обслуживание клиентов.

3.6 Работа и апробация веб-приложения

Для того чтобы убедиться, что проект был создан правильно. Введите команду runserver: (env) python manage.py runserver Performing system checks... System check identified no issues (0 silenced). Django, using settings Starting development server at http://127.0.0.1:8000/ Quit the server with CONTROL-C. Django запускает сервер, чтобы вы могли просмотреть проект в своей системе и проверить, как он работает. Когда вы запрашиваете страницу, вводя URL в браузере, сервер Django отвечает на запрос; для этого он строит соответствующую страницу и отправляет страницу браузеру. В точке Django проверяет правильность созданного проекта; в точке выводится версия Django и имя используемого файла настроек; в точке возвращается URL-адрес, по которому доступен проект. URL http://127.0.0.1:8000/ означает, что проект ведет

прослушивание запросов на порте 8000 локального хоста (localhost), то есть вашего компьютера. Термином «локальный хост» обозначается сервер, который обрабатывает только запросы вашей системы; он не позволяет никому другому просмотреть разрабатываемые страницы. Теперь откройте браузер и введите URL http://localhost:8000/ — или http://127.0.0.1:8000/, если первый адрес не работает. Пока не завершайте работу сервера (но, когда вы захотите прервать ее, это можно сделать нажатием клавиш Ctrl+C).

ЗАКЛЮЧЕНИЕ

Разработанная в ходе выполнения дипломной работы информационная система «Гостиничного комплекса», является актуальной на сегодняшний день и имеет большую практическую значимость.

В результате выполнения дипломной работы были решены задачи, поставленные в начале работы. Была разработана структура базы данных; в программу были включены функции поиска, выполнения различных запросов. При этом были учтены все требования, выдвинутые в начале выполнения данной работы. Для формирования сильной клиентской базы необходимо учитывать важные нюансы. Другими словами, нам нужно сосредоточиться на потребностях и пожеланиях наших клиентов. Из этого следует вывод, что основа данного метода управления бизнесом - это клиент. Рентабельность и успех бизнеса зависят от скорости, с которой компании реагируют на меняющиеся рыночные условия. На сегодняшний день ситуация на рынке Казахстана неоднозначна. Отечественные предприятия, гоняясь за прибылью и повышая свою конкурентоспособность, изобретают все более модернизированные и продуктивные методы поиска клиентов. Для того чтобы сформировать крепкую клиентскую базу следует учесть важный нюанс, а именно, необходимо ориентироваться на нужды и желания клиента. Отсюда становится понятным, что основа данного метода управления бизнесом - это клиент. Разработанная программа устойчиво выполняет все свои функции.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Welling L., Thomson L. Python and MySQL Web Development, 4-edition, 2009 англ.,обучающийдиск.

2 Безруков Н.Н. Компьютерные вирусы. - М.: Наука, 2014 г, - 660 с

3 БьюлиА.. Изучаем SQL, pdf, 2017, - 110 с.

4 Веллинг Л., Томсон Л. MySQL. Учебное пособие, pdf, 2015, - 113 с.

5 Веллинг Л., Томсон Л. Разработка Web-приложений с помощью PHP и MySQL, 3-е издание, djvu, 2008, - 97 с.

6 Грабер М. SQL, djvu, 2009, - 455 с.

7 Грабер М. Введение в SQL, pdf, - 256 с.

8 Дронов В.А. JavaScript в Web-дизайне. - СПб.: БХВ-Петербург, 2001. Бейли Л. Изучаем SQL. O'REILLY, pdf, 2012, - 98 с.

9 Дэвис М., ФиллипсД.. Изучаем PHP и MySQL, pdf, 2008, - 312 с.

10 Дюбуа П. MySQL, djvu, 2004, - 165 с.

11 Зандстра М. PHP объекты, шаблоны и методики программирования, djvu, 2010, - 288 с.

12 Калина А.В. Организация и оплата труда в условиях рынка (аспект эффективности): Учебно-методическое пособие. - К.: МАУП, 1997. - 300 с.

13 Котеров Д.В. Html в подлиннике, djvu, 2010, - 216 с.

14 Кузнецов М. Практика создания веб-сайтов, pdf + CDROM, 2009,- 200 с.

15 Кузнецов М., Симдянов И. MySQL 5.0, djvu, 2010, - 150 с.

16 Леонтьев Б.В. Web-Дизайн: Тонкости, хитрости и секреты / Леонтьев Б.В. - М. Майор, 2011, -170 с..

17 Никсон Р. Создаем динамические веб сайты с помощью PHP MySQL и JavaScript, djvu, 2011, - 74 с.

18 Петюшкин А. B.html Экспресс - курс / Петюшкин А.В. - СПБ.: БХВ-Петербург, 2013, - 256 с. ил.

19 Программное обеспечение. Энциклопедия ПК. Выпуск 2. Обучающий диск,

20 Руководство по PHP, chm, 2014, обучающий диск.

21 Стейнмец У., ВардБ.. Самоучитель 75 готовых решений для вашего сайта на PHP, pdf, 2009, - 270 с.

22 Строганов А.С. Ваш первый сайт с использованием PHP-скриптов, djvu, 2015, - 348 с.

23 Суэринг С., Конверс Т., Парк Д. PHP и MySQL. Библия программиста, djvu, 2010, - 111с.

24 Титоренко Г.А. Информационные технологии управления. - М.: ЮНИТИ-ДАНА, 2003 г, - 180 с.

25 Хольцнер С. Python в примерах, djvu, 2010, - 215 с.

Приложение А

Листинг модуля для регистраций

```
{% extends 'base.html' %}
{% block content-title %}
<h1>Войтивсистему</h1>
{% endblock %}
{% block login-status %}
{% endblock %}
{% block main-content %}
{% ifform.errors %}
<p class="notMatched">Имя пользователя и пароль не совпадают. Пожалуйста, попробуйте еще раз.</p>
{% endif %}
  {% if next %}
    {% if user.is_authenticated %}
<p>У вас нет прав доступа к этой странице.</p>
    {% else %}
<p>Вы должны войти в систему, чтобы просмотреть эту страницу.</p>
{% endif %}
  {% endif %}
<form method="post" action="{% url 'login' %}">
    {% csrf_token %}
<div class="form-row align-items-center">
<div class="form-group col-auto">
<label for="{{ form.username.id_for_label }}">имяпользователя</label>{{ form.username }}
</div>
<div class="form-group col-auto">
<label for="{{ form.password.id_for_label }}">пароль</label>{{ form.password }}
</div>
</div>
<input type="submit" value="Login"/>
<input type="hidden" name="next" value="{{ next }}"/>
</form>

  {# URLconfвпредставлениирassword_reset #}
<p><a href="{% url 'password_reset' %}">Забылипароль?</a></p>

{% endblock %}
```

Продолжение приложения А

Листинг модуля для домашней страницы

```
{% load i18n %}
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="This is internal hotel management system">
<meta name="author" content="Software Engineers' Group">
<!--<link rel="icon" href="../../../../favicon.ico"> -->
<!-- Title -------------- -->
<title>{{ title }}</title>
<!-- --------------------- -->
<!-- Bootstrap core CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
<link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<!-- --------------Custom styles------------------------------- -->
{% load static %}
<link href="{% static 'css/styles.css' %}" rel="stylesheet">
<!-- ------------------------------------------------------------ -->
</head>

<body>
<!-- --------------------------Header ----------------------------------- -->
<header>
<nav class="navbarnavbar-expand-md navbar-dark fixed-top bg-dark">
<!-- --------------Navigation Bar--------------------- -->
<a class="navbar-brand" href="{% url 'index' %}">
```

Продолжение приложения А

```
<imgsrc="{% static "/img/hotel-fav-icon.png" %}" style="width:50px; height=50px; padding:5px;">Гостиничныйкомплекс</a>
<button class="navbar-toggler d-lg-none" type="button" data-toggle="collapse"
        data-target="#top_navbar" aria-controls="top_navbar" aria-expanded="false"
        aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<!-- ------------------------------------------------------------ -->
<div class="collapse navbar-collapse justify-content-end" id="top_navbar">
<ul class="navbar-nav">
<!-- -------------------Navigation Items--------------- -->
        {% block navbar-items %}
          {% if user.is_authenticated %}
<li class="nav-item">
<a class="nav-link" href="{% url 'profile' %}">Профиль</a>
</li>
          {% endif %}
<li class="nav-item">
<a class="nav-link" href="#">помощь</a>
</li>
        {% endblock %}
<!-- ---------------------------------------------------- -->
</ul>
</div>
</nav>
</header>
<!-- ------------------------------------------------------------------ -->

<div class="container-fluid">
<div class="row">
    {% block sidebar %}
<!-- ---------------------Sidebar Navigation------------------------- -->
<nav class="col-sm-3 col-md-2 d-none d-sm-block bg-light sidebar">
<!-- --------------Sidebar Items------------------ -->
        {% block sidebar-items %}
<ul class="navnav-pills flex-column">
<li class="nav-item">
<a class="nav-link" href="{% url 'index' %}">Статусотеля</a>
</li>
```

Продолжение приложения А

```
<li class="nav-item">
<a class="nav-link" href="{% url 'guest-list' %}">гости</a>
</li>
<li class="nav-item">
<a class="nav-link" href="{% url 'rooms' %}">комнаты</a>
</li>
<li class="nav-item">
<a class="nav-link" href="{% url 'payment-index' %}">платежи</a>
</li>
</ul>
                {% if user.is_authenticated %}
<ul class="navnav-pills flex-column">
<li class="nav-item">
<a class="nav-link" href="{% url 'reserve' %}">резерв</a>
</li>
</ul>
<ul class="navnav-pills flex-column">
<li class="nav-item">
<a class="nav-link" href="{% url 'reservations' %}">Бронирование</a>
</li>
<li class="nav-item">
<a class="nav-link" href="{% url 'check_in-list' %}">регистрация</a>
</li>
<li class="nav-item">
<a class="nav-link" href="{% url 'check_out-list' %}">релиз</a>
</li>
</ul>
                {% endif %}
<ul class="navnav-pills flex-column">
<li class="nav-item">
<a class="nav-link" href="#">Отзывы</a>
</li>
</ul>
<ul class="navnav-pills flex-column">
                    {% block login-status %}
                       {% if user.is_authenticated %}
<li class="nav-item">
<a class="nav-link" href="#">{{ user.first_name }} {{ user.last_name }}</a>
<a class="nav-link" href="{% url 'logout' %}?next={{ request.path }}">выход</a>
</li>
```

Продолжение приложения А

```
        {% else %}
        <li class="nav-item">
        <a class="nav-link" href="{% url 'login' %}?next={{ request.path
}}">Войтивсистему</a>
        </li>
        <li class="nav-item">
        <a class="nav-link" href="{% url 'signup' %}?next={{ request.path
}}">Вынезарегестрированы?</a>
        </li>
                        {% endif %}
                    {% endblock %}
        </ul>
                {% endblock %}
        <!-- ------------------------------------------------------------- -->
        </nav>
            {% endblock %}
        <!-- ------------------------------------------------------------------------- -->

        <!-- -------------------------Main Body Content---------------------- -->
        <main role="main" class="col-sm-9 ml-sm-auto col-md-10 pt-3"
                style="background: linear-gradient(to right, lightblue, lightyellow)">
        <!-- ---------Content Heading----------- -->
            {% block content-title %}
        <h1>{{ title }}</h1>
            {% endblock %}
        <!-- ------------------------------------ -->

        <!-- Content for the heading previously mentioned -->
            {% block main-content %}
        <section class="row text-center placeholders">
        </section>
            {% endblock %}
        <!-- ------------------------------------------------- -->

        <!-- --------------Paginator--------------- -->
            {% block pagination %}
        <!-- Only shows paginator if the page is paginated -->
                {% if is_paginated %}
        <nav aria-label="Page Navigation">
        <ul class="pagination">
        <!-- Only shows previous button when it has previous -->
```

Продолжение приложения А

```
{% if page_obj.has_previous %}
<!-- Code for previous page -->
<!-- {{ page_obj.previous_page_number }} is previous page number. -->
<li class="page-item"><a class="page-link"
href="{{ request.path }}?page={{ page_obj.previous_page_number
}}">передний</a>
</li>
                    {% endif %}
<!-- Pagination Numbers -->
                    {% for page_no in page_obj.paginator.page_range %}
<li class="page-item {% if page_no == page_obj.number %} active {% endif
%}"><a
                              class="page-link"
href="{{ request.path }}?page={{ page_no }}">{{ page_no }}</a></li>
                    {% endfor %}
<!-- ------------------ --->
<!-- Only shows next button if it has next page -->
                    {% if page_obj.has_next %}
<!-- {{ page_obj.next_page_number }} is next page. -->
<li class="page-item"><a class="page-link"
href="{{ request.path }}?page={{ page_obj.next_page_number }}">после</a>
</li>
                    {% endif %}
</ul>
</nav>
            {% endif %}
<!-- ------------------------------------------------------- -->
        {% endblock %}
</main>
<!-- ------------------------------------------------------------------------------------->
</div>
</div>

<!-- Bootstrap core JavaScript
=================================================== -->
<!-- Placed at the end of the document so the pages load faster -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.jintegrit
y="sha384-
```

Продолжение приложения А

```
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusv
fa0b4Q"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/js/bootstrap.min.js"
integrity="sha384-
a5N7Y/aK3qNeh15eJKGWxsqtnX/wWdSZSKp+81YjTmS15nvnvxKHuzaWwXHD
li+4"
crossorigin="anonymous"></script>
<script src="https://use.fontawesome.com/c76346d885.js"></script>
{% load static %}
<script src="{% static 'js/script.js' %}"></script>
{% block media-files %}
{% endblock %}
<script>
$(window).on("load", function () {
if ($("form")) {
$("input[type!='submit']").addClass("form-control");
$("input[type='submit']").addClass("btnbtn-primary")
}
});
</script>
{% block additional-javascript %}
{% endblock %}
</body>
</html>

{% extends 'base.html' %}
{% block content-title %}
<h1>{{ title }}</h1>
{% endblock %}
<!--ToDo Work on FrontEnd -->
{% block main-content %}
<div class="row">
<div class="card .col-md-3">
<div class="card-body">
<div class="card-header bg-primary text-
white">Количествоспальныхмест:</div>
<div class="card-body bg-dark text-white">{{ total_num_rooms }}</div>
</div>
</div>
```

Продолжение приложения А

```
<div class="card .col-md-3">
<div class="card-body">
<div class="card-header bg-primary text-white">Пустыеномера</div>
<div class="card-body bg-dark text-white">{{ available_num_rooms }}</div>
</div>
</div>

<div class="card .col-md-3">
<div class="card-body">
<div class="card-header bg-primary text-white">Забронированныеномера:</div>
<div class="card-body bg-dark text-white">{{ total_num_reservations }}</div>
</div>
</div>

<div class="card .col-md-3">
<div class="card-body">
<div class="card-header bg-primary text-white">Общееколичествосотрудников:</div>
<div class="card-body bg-dark text-white">  {{ total_num_staffs }} </div>
</div>
</div>

<div class="card .col-md-3">
<div class="card-body">
<div class="card-header bg-primary text-white">Всегоклиентов:</div>
<div class="card-body bg-dark text-white">{{ total_num_customers }}</div>
</div>
</div>
    {% if user.is_authenticated %}
<div class="card .col-md-3">
<div class="card-body">
<div class="card-header bg-primary text-white">Последнееебронирование:</div>
        {% if last_reserved_by.reservation_id %}
<div class="card-body bg-dark text-white">Бронирование: <a
href="{% url 'customer-detail' last_reserved_by.customer.customer_id %}" class="text-white">{{ last_reserved_by.customer }}</a>
</div>
<div class="card-body bg-primary text-white">
```

Продолжение приложения А

```
        {% if perms.main.can_view_staff_detail %}
```

```
<a href="{% url 'staff-detail' last_reserved_by.staff.staff_id %}" class="text-white">{{ last_reserved_by.staff }}</a>
                                {% else %}{{ last_reserved_by.staff }}
                            {% endif %}
</div>
                {% else %}
<div class="card-body bg-dark text-white">Ещенет</div>
                {% endif %}
</div>
</div>
        {% endif %}
</div>
{% endblock %}
from django.contrib.auth.decorators import permission_required
from django.contrib.auth.mixins import PermissionRequiredMixin
from django.contrib.auth.models import User, Group
from django.core.exceptions import ValidationError
from django.db import transaction, IntegrityError
from django.db.models import Q
from django.http import Http404
from django.shortcuts import render, redirect, get_object_or_404 # Үлгідекөрсетуүшін
from django.urls import reverse_lazy
from django.utils import timezone
from django.utils.translation import ugettext_lazy as _
from django.views import generic

from .forms import Signup, ReservationForm, CheckInRequestForm
from .models import Room, Reservation, Customer, Staff  # Import Models

def index(request):
    """
Этовидшедевра.
    Эта функция основана на представлении.
    """
page_title = _("Дипломный проект Көркем") # Беттіңтақырыбыжәнетақырыпүшін
total_num_rooms = Room.objects.all().count()
available_num_rooms = Room.objects.exclude(reservation__isnull=False).count()
```

Продолжение приложения А

```
total_num_reservations = Reservation.objects.all().count()
```

```
total_num_staffs = Staff.objects.all().count()
total_num_customers = Customer.objects.all().count()
    if total_num_reservations == 0:
last_reserved_by = Reservation.objects.none()
    else:
last_reserved_by =
Reservation.objects.get_queryset().latest('reservation_date_time')

    return render(
        request,
        'index.html',
        # контекстүлгігежіберілген.
        # үлгілердеайнымалыретіндепайдаланылады
# қайжердеболса, келесіфункцияныңайнымалысыболады
{
            'title': page_title,
            'total_num_rooms': total_num_rooms,
            'available_num_rooms': available_num_rooms,
            'total_num_reservations': total_num_reservations,
            'total_num_staffs': total_num_staffs,
            'total_num_customers': total_num_customers,
            'last_reserved_by': last_reserved_by,
        }
    )


@transaction.atomic
def signup(request):
    title = "Зарегистрироваться"
    if request.user.is_authenticated:
request.session.flush()
    if request.method == 'POST':
        form = Signup(request.POST)
        if form.is_valid():
            try:
                with transaction.atomic():
staffs_group = get_object_or_404(Group, name__iexact="Staffs")
form.save()
staff_id = form.cleaned_data['staff_id']
                    username = form.cleaned_data['username']
```

Продолжение приложения А

```
s = get_object_or_404(Staff, staff_id__exact=staff_id)
```

```
s.user = get_object_or_404(User, username__iexact=username)
s.user.set_password(form.cleaned_data['password1'])
s.user.groups.add(staffs_group)
s.user.save()
s.save()
        except IntegrityError:
            raise Http404
        return redirect('index')
    else:
        form = Signup()

    return render(
        request,
        'signup.html', {
            'form': form, 'title': title},
    )

@permission_required('main.add_reservation', 'login', raise_exception=True)
@transaction.atomic
def reserve(request):
    title = "Включитьбронирование"
    reservation = Reservation.objects.none()
    if request.method == 'POST':
reservation_form = ReservationForm(request.POST)
        if reservation_form.is_valid():
            try:
                with transaction.atomic():
                    customer = Customer(
first_name=reservation_form.cleaned_data.get('first_name'),
middle_name=reservation_form.cleaned_data.get('middle_name'),
last_name=reservation_form.cleaned_data.get('last_name'),
email_address=reservation_form.cleaned_data.get('email'),
contact_no=reservation_form.cleaned_data.get('contact_no'),
                        address=reservation_form.cleaned_data.get('address'),
                    )
customer.save()
                    staff = request.user
                    reservation = Reservation(
                        staff=get_object_or_404(Staff, user=staff),
                        customer=customer,
```

Продолжение приложения А

```
no_of_children=reservation_form.cleaned_data.get('no_of_children'),
```

```
        no_of_adults=reservation_form.cleaned_data.get('no_of_adults'),

expected_arrival_date_time=reservation_form.cleaned_data.get('expected_arrival_date_time'),

expected_departure_date_time=reservation_form.cleaned_data.get('expected_departure_date_time'),
        reservation_date_time=timezone.now(),
                        )
        reservation.save()
                        for room in reservation_form.cleaned_data.get('rooms'):
        room.reservation = reservation
        room.save()
                except IntegrityError:
                    raise Http404
                return render(
                    request,
                    'reserve_success.html', {
                        'reservation': reservation,
                    }
                )
            else:
        reservation_form = ReservationForm()

            return render(
                request,
                'reserve.html', {
                    'title': title,
                    'reservation_form': reservation_form,
                }
            )


        defreserve_success(request):
            pass


        # Generic ListViewнемесеDetailViewүшін, әдепкіүлгілердіүлгілерде / {{app_name}} / {{template_name}}
        # Әдепкібойыншаtemplate_name = modelName_list || modelName_detail.
```

Продолжение приложения А

```
        # мысалыroom_list, room_detail
```

```
# @ permission_required ('main.can_view_staff')
class RoomListView(PermissionRequiredMixin, generic.ListView):
"""
Посмотреть список номеров.
Создает общий элемент списка.
    """
model = Room  # Листинг объектілерінмоделінтаңдайды
paginate_by = 5  #Бұлқаншаобъектілердіңкөмегіменжасалуыкерек
title = _("Список номеров") #
Бұлтақырыпжәнетақырыпүшінпайдаланылады
permission_required = 'main.can_view_room'

# Әдепкідетекүлгінысандарыконтекстретіндежіберіледі
# Дегенмен, қосымшамәтінмәнextra_contextөрісіарқылыберілуімүмкін
# Мұндатақырыпатауыберілді.
extra_context = {'title': title}

# Әдепкібойынша:
# template_name = room_list
# егероныөзгерткіңізкелсе, өрісүлгісінқолданыңыз
# мұндамұныжасамаңыз, себебіоләдепкібойыншаорындалады.
# өзкөзқарастарыңызүшіноныжасайаласыз.

defget_queryset(self):
filter_value = self.request.GET.get('filter', 'all')
    if filter_value == 'all':
filter_value = 0
eliffilter_value == 'avail':
filter_value = 1
    try:
new_context = Room.objects.filter(availability__in=[filter_value, 1])
exceptValidationError:
raise Http404(_("Ложный фильтр доказательство предоставляется."))
return new_context

defget_context_data(self, **kwargs):
    context = super(RoomListView, self).get_context_data(**kwargs)
    context['filter'] = self.request.GET.get('filter', 'all')
    return context
```

Продолжение приложения А

```
class RoomDetailView(PermissionRequiredMixin, generic.DetailView):
```

```
"""
Посмотреть детали комнаты
Создает полнофункциональный вид
"""
    # The remaining are same as previous.
model = Room
    title = _("Информацияономере")
permission_required = 'main.can_view_room'
extra_context = {'title': title}


class ReservationListView(PermissionRequiredMixin, generic.ListView, generic.FormView):
"""
Просмотр списка покупок.
Создает общий элемент списка.
    """
model = Reservation
    # querysetөрісісұрауарқылыкөрсетілетіннысандардытаңдайды.
    #
Мұнданысандаркемутәртібіменрезервтеукүнініңуақытыменкөрсетіледі
queryset = Reservation.objects.all().order_by('-reservation_date_time')
    title = _("Списокбронирования")
paginate_by = 5
allow_empty = True
form_class = CheckInRequestForm
success_url = reverse_lazy('check_in-list')
permission_required = 'main.can_view_reservation'
extra_context = {'title': title}

    @transaction.atomic
defform_valid(self, form):
    try:
        with transaction.atomic():
checkin = form.save(commit=False)
checkin.user = self.request.user
checkin.save()
    except IntegrityError:
        raise Http404
permission_required = 'main.can_view_reservation'
```

Продолжение приложения А

```
return super().form_valid(form)
```

```
class ReservationDetailView(PermissionRequiredMixin, generic.DetailView):
"""
Смотрите детали вашего бронирования
Создает полнофункциональный вид
"""
    model = Reservation
    title = _("Информацияобронировании")
permission_required = 'main.can_view_reservation'
raise_exception = True
extra_context = {'title': title}


class CustomerDetailView(PermissionRequiredMixin, generic.DetailView):
"""
Посмотреть подробную информацию о клиенте
Создает полнофункциональный вид
"""
    model = Customer
    title = _("Информациядляклиентов")
permission_required = 'main.can_view_customer'
raise_exception = True
extra_context = {'title': title}


class StaffDetailView(PermissionRequiredMixin, generic.DetailView):
"""
Больше информации о сотрудниках
Создает полнофункциональный вид
"""
    model = Staff
    title = _("Информацияоперсонале")
permission_required = 'main.can_view_staff_detail'
extra_context = {'title': title}


class ProfileView(generic.TemplateView):
template_name = 'profile.html'
    title = "Профиль"
```

Продолжение приложения А

```
extra_context = {'title': title}
```

```
defget_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        if self.request.user.is_authenticated:
            context['information']                =                get_object_or_404(Staff,
user=self.request.user)
            context['user_information'] = self.request.user
else:
raise Http404("Вы не авторизованы")
return context
class GuestListView(PermissionRequiredMixin, generic.ListView):
"""
Смотрите список гостей отеля.
    """
model = Customer
paginate_by = 5
allow_empty = True
queryset = Customer.objects.all().filter(Q(reservation__checkin__isnull=False),
                                         Q(reservation__checkin__checkout__isnull=True))
permission_required = 'main.can_view_customer'
template_name = 'main/guest_list.html'
    title = 'Списокгостей'
context_object_name = 'guest_list'
extra_context = {'title': title}
```

Приложение Б

Интерфейс программы

После запуска программы откроется окно приложения на вкладке «Войти в систему»

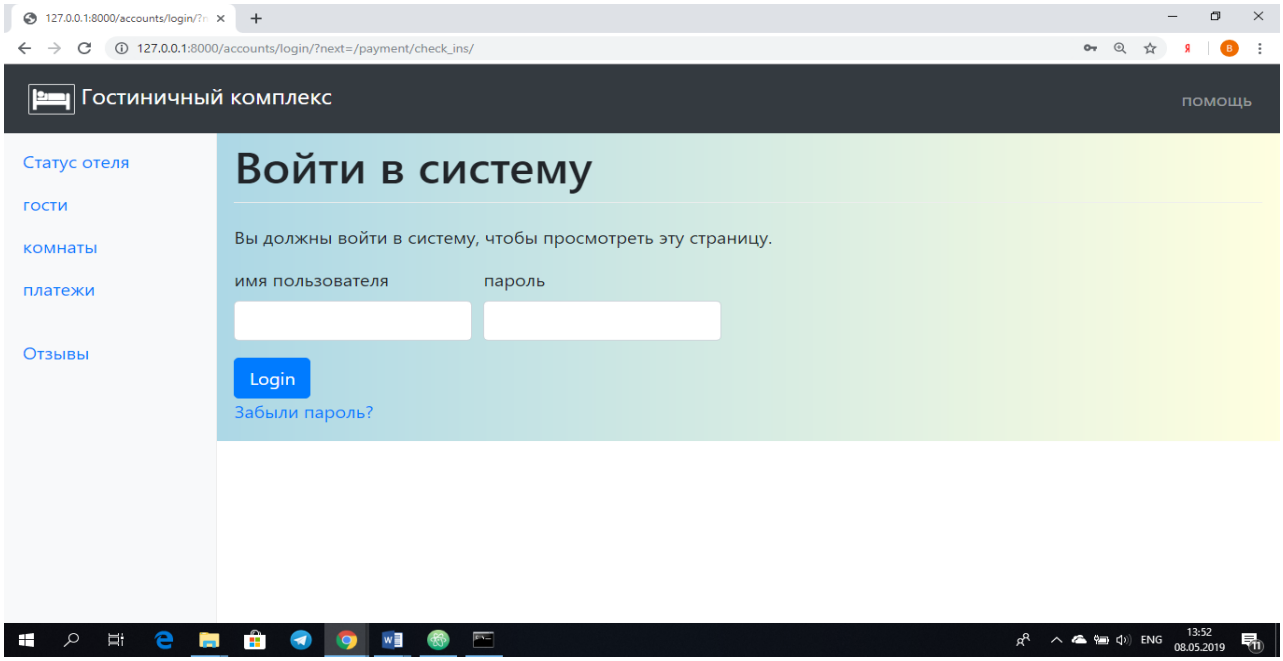

Рисунок Б.1 – Войти в систему

Вкладка «Номера» содержит основные сведения о гостиничных номерах

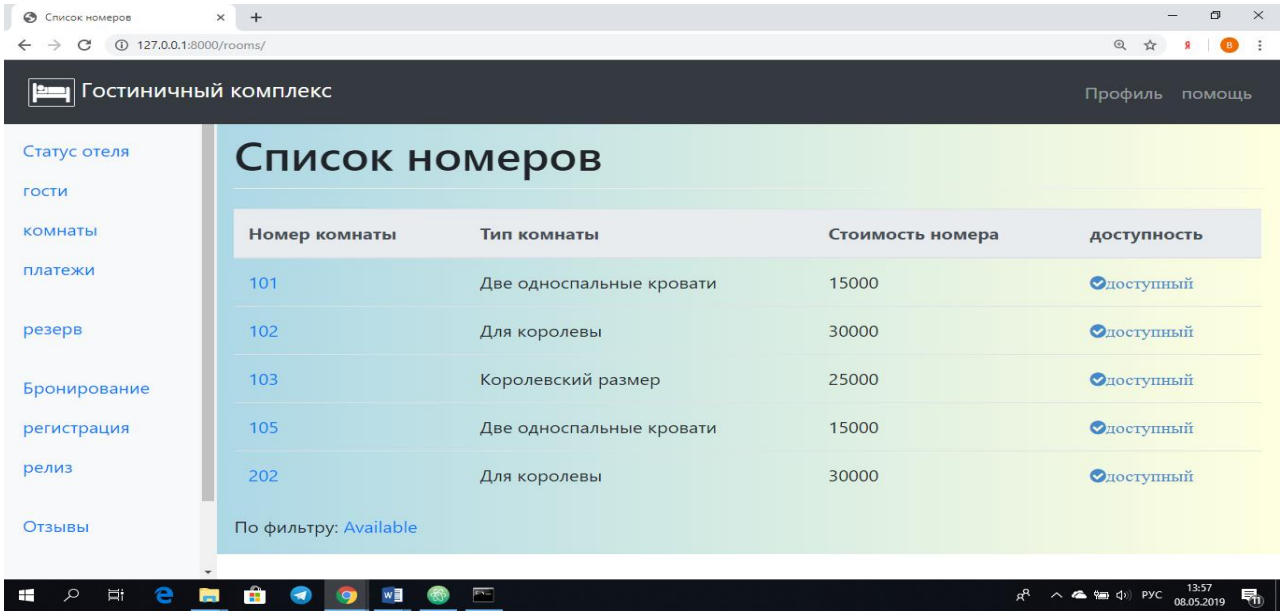

Рисунок Б.2 – Номера

Продолжение приложения Б

Вкладка «Клиенты» содержит сведения о клиентах гостиницы

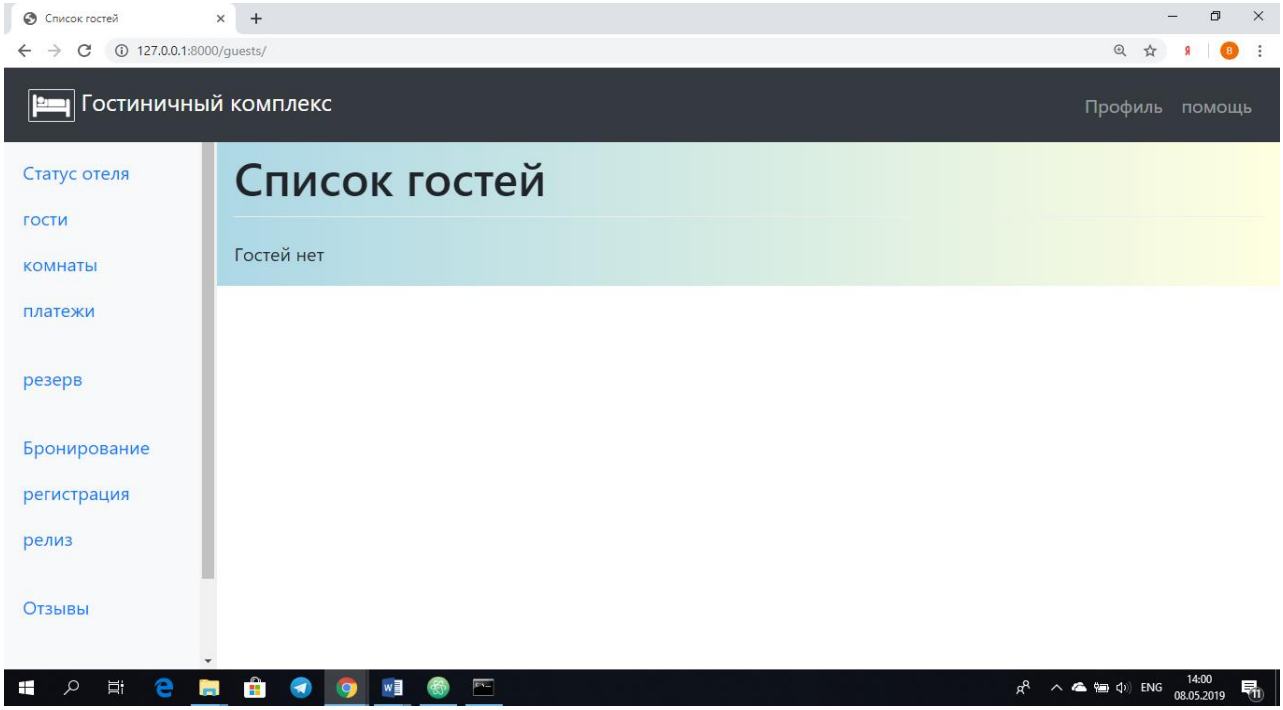

Рисунок Б.3 – Клиенты

Содержимое вкладки «Статус отеля» разделено на две части: в первой части можно просмотреть свободные номера, во второй – занятые

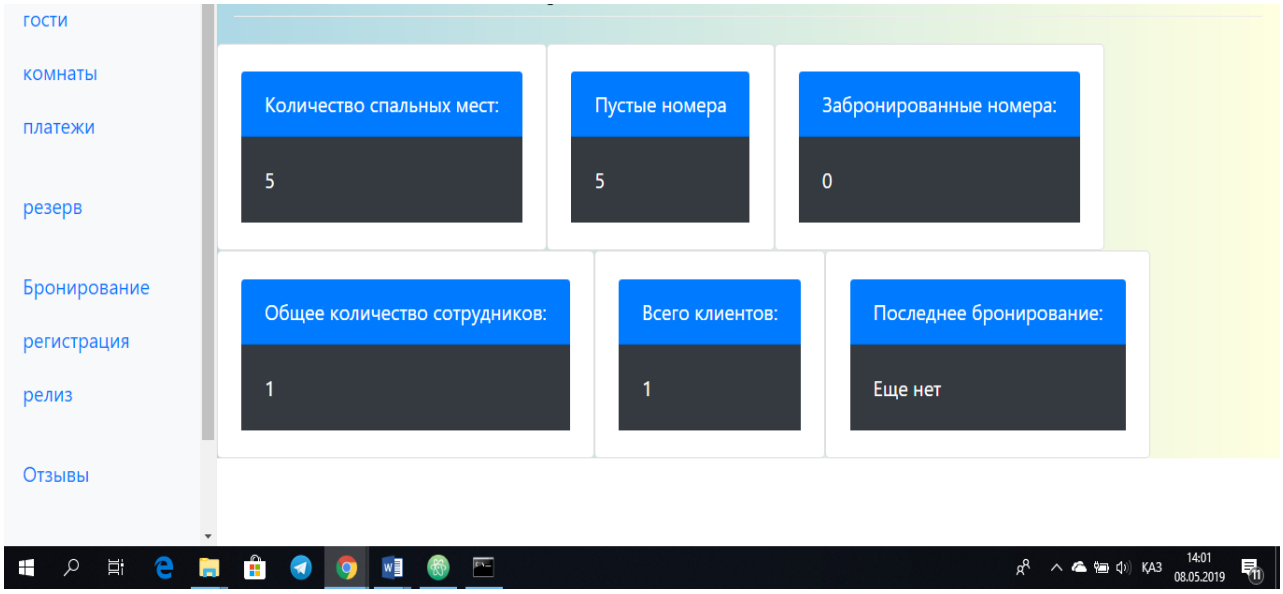

Рисунок Б.4 – Статус отеля

Продолжение приложения Б

Нажатие на вкладку «Платежи» позволяет просмотреть более подробную информацию о номерах и клиентах

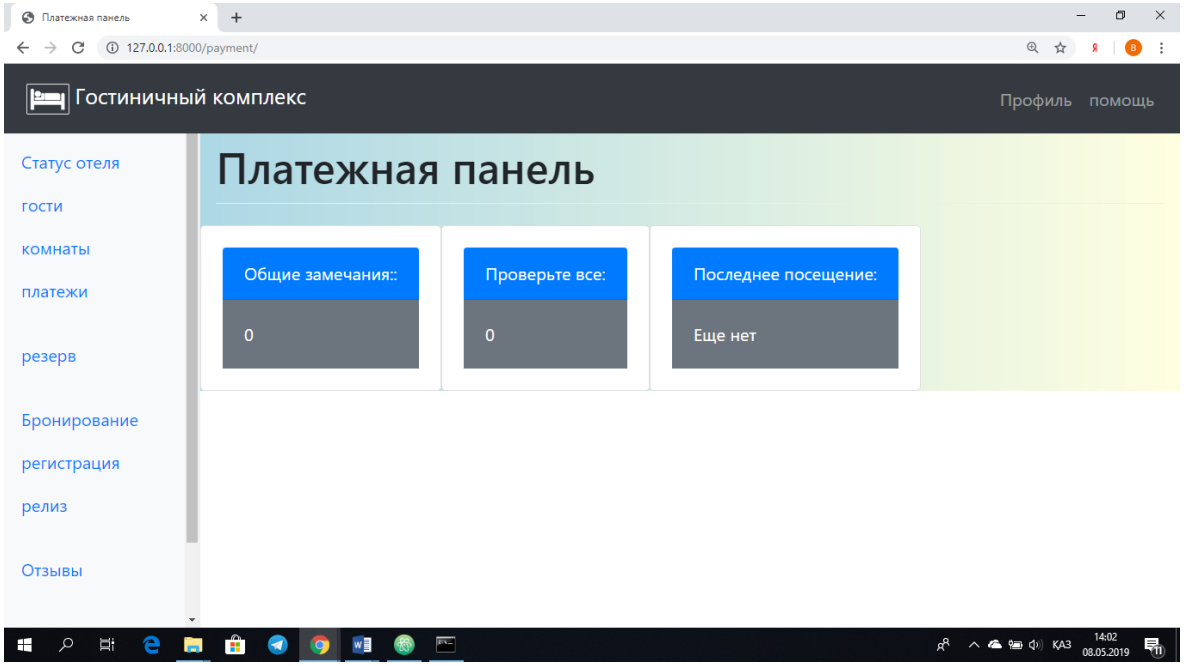

Рисунок Б.5 – Платежи

На вкладке «Регистрация» выбирается гостиничный номер и фамилия, имя, отчество клиента, а также указывается период проживания в гостинице

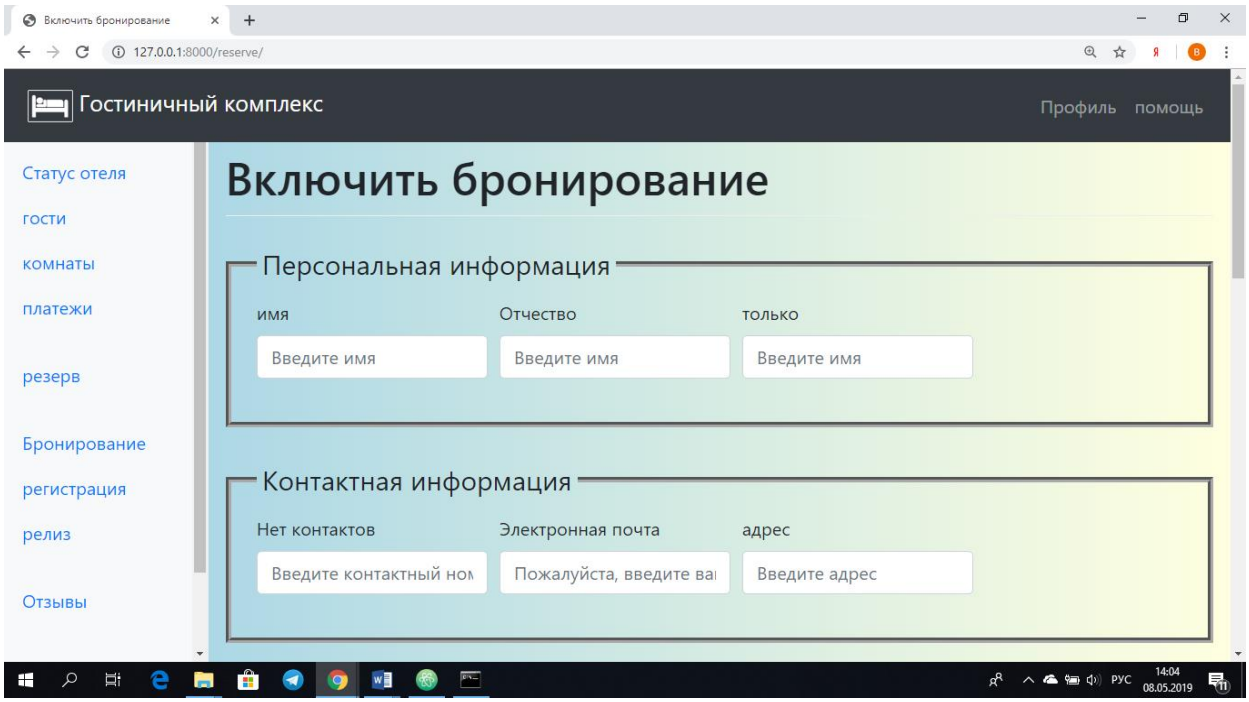

Рисунок Б.6 – Регистрация

Продолжение приложения Б

При нажатии на кнопку «Бронирование» откроется отчет

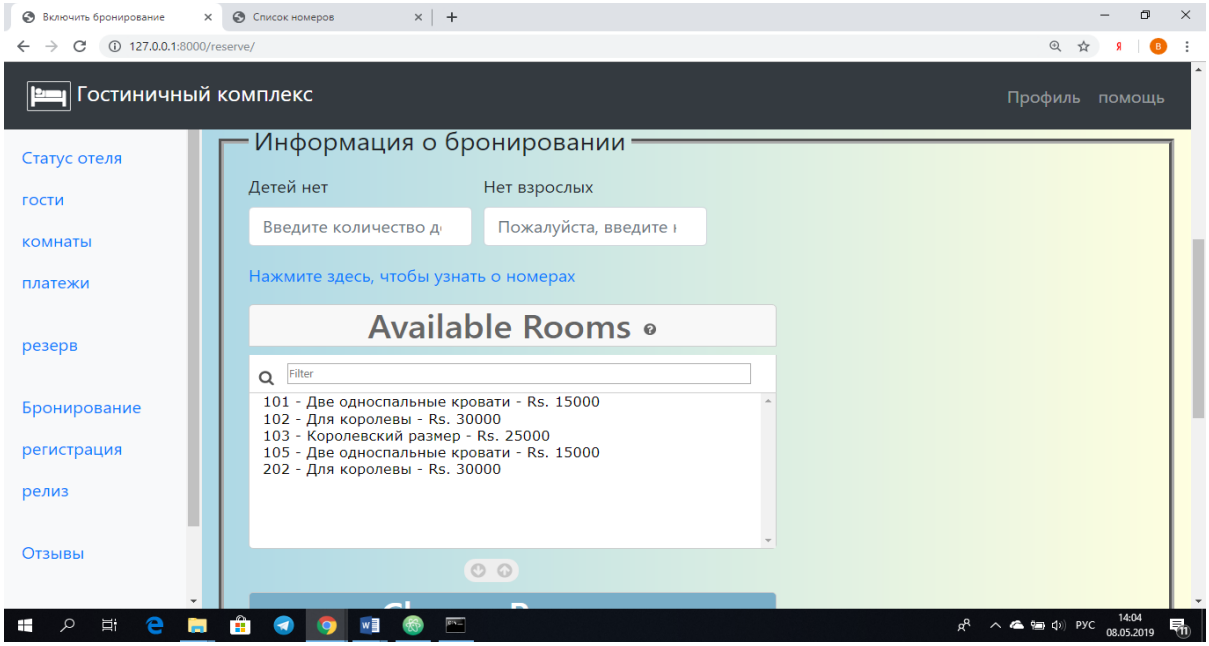

Рисунок Б.7 – Бронирование

При добавлении новой или изменении текущей записи кнопки «Добавить» и «Изменить» меняются на «Сохранить» и «Отменить». Кнопка «Удалить» и кнопки навигации становятся недоступными

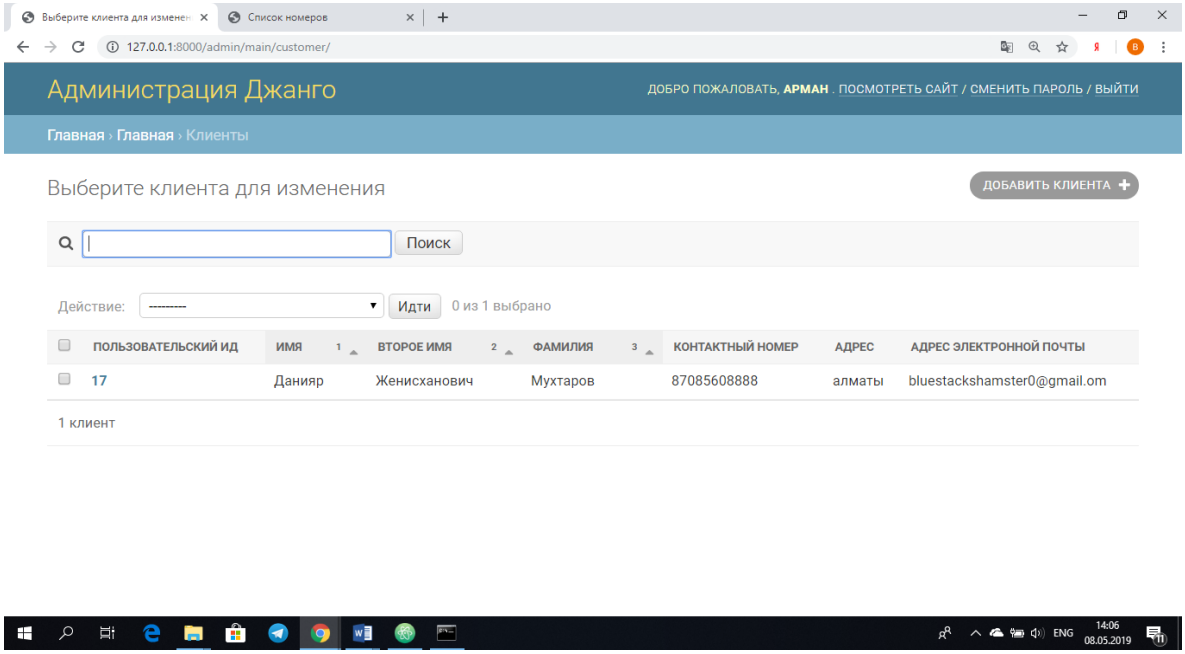

Рисунок Б.8 – Кнопки добавления, изменения и удаления записи

Продолжение приложения Б

При удалении текущей записи откроется форма-подтверждение на удаление

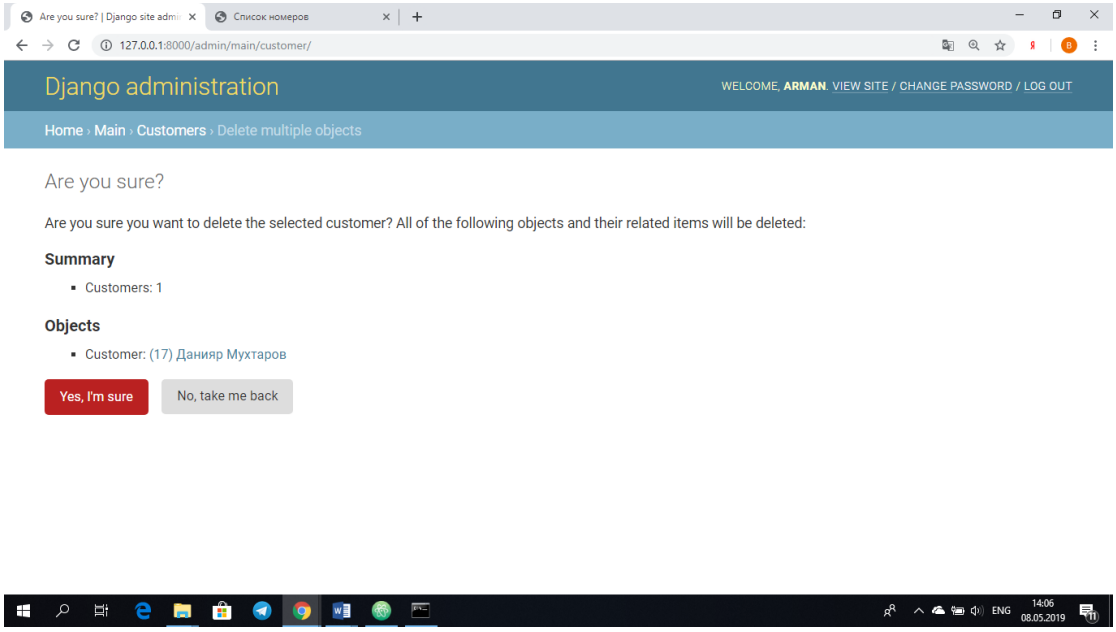

Рисунок Б.9 – Подтверждение на удаление