СОДЕРЖАНИЕ

ВВЕДЕНИЕ……………………………………………………………………………………. 6

1 АНАЛИЗ И КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 8

2 ПОСТАНОВКА ЗАДАЧИ…………………………………………………………………... 11

3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ…………………………………………………… 13

3.1 UML-моделирование предметной области………………………………………. 13

3.2 Построение UML диаграмм реализаций…………………………………………. 17

4 ОПИСАНИЕ ПРОГРАММЫ……………………………………………………………….. 21

4.1 Общие сведенья …………………………………………………………………… 21

4.2 Вызов и загрузка…………………………………………………………………… 21

4.3 Назначение и логическая структура……………………………………………… 22

4.4 Описание физической модели базы данных…………………………………….. 23

4.5 Описание программной реализации……………………………………………… 26

4.6 Описание выбранных технологий……………………………………………….. 29

5 КОНКУРЕНТЫ……………………………………………………………………………… 31

ВЫВОДЫ………………………………………………………………………………………. 32

ПЕРЕЧЕНЬ ССЫЛОК………………………………………………………………………… 33

ВВЕДЕНИЕ

Областью применения данного курсового проекта является туристический сектор. Основными пользователями есть туристы и люди, которые просто любят путешествовать.

Проблема этой области состоит в том, что на данный момент некоторые люди боятся путушествовать из-за того, что много не знают о месте, куда они хотят поехать, не используя новые технологии, которые могут облегчить процессы обмена и хранения информации.

Использование базы данных позволит более удобно обмениваться информацией и хранить ее. База данных может хранить в себе информацию о клиентах, вопросах, ответах, горотдах, странах. При этом она сможет предоставить удобный доступ к этой информации. Поиск, фильтрацию и сортировку данных.

Программная система сможет обеспечить пользователя часто необходимой информацией. Для того, чтобы узнать, какие какую-то информацию о городе, не нужно будет просматривать все страницы об этом городе в Интернете, или обзванивать всех знакомых. Можно просто добавить вопрос на сайт и другие пользователи смогут ответить на этот вопрос. Также она может предложить ответить на существующий вопрос или просмотреть информацию о прошлых вопросах.

Хранение всех данных в одной базе обеспечит целостность информации. Если информация про один вопрос поменялась, не будет необходимости проверять все места, где информация о нем была скопирована. Достаточно будет один раз исправить эту информацию, а в остальных местах система автоматически изменит данные на более актуальные.

Когда создается маршрут, который проходит через несколько мест, можно выбирать места, которые ранее уже использовались в программе. Если информация о месте изменится, то программа скорректирует информацию о маршруте.

Из приведенной оценки состояния предметной области можно сделать вывод об актуальности разрабатываемой программы. Она существенно облегчит деятельность людей в сфере туристического бизнеса, выполняя за них большинство повседневных задач.

Таким образом целью данного курсового проекта есть спроектировать базу данных, выбрать и использовать СУБД для работы с данными, которые лежат в базе, а также разработать программную систему, которая позволит пользователю взаимодействовать с данными через удобный интерфейс программы. Еще целью является создание хорошо спроектированной системы, которая будет масштабируемой и безопасной.

В качестве СУБД была выбрана MySQL из-за ее быстродействия, наличия бесплатных лицензий и поддержки параллельных запросов. Для реализации информационной системы будет использован Spring Framework, AngularJS. Эта технология позволяет довольно просто создавать приложения с удобным и простым интерфейсом. В качестве интегрированной среды разработки была выбрана InteliJ IDEA из-за встроенной поддержки Java и MySQL.

1 АНАЛИЗ И КОНЦЕПТУЛЬАНОЕ ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Исследуемая система представляет собой туристическую сектор, которая организовывает походы для туристов разных уровней. Клиент может отправлятся в какое-то путешествие. После этого туристу будет предложено заполнить заявление, с указанием информации о себе и документах, необходимых для регистрации в спасательных. У каждого похода есть свой инструктор, который поведет группу. Заранее известна дата начала похода, его длительность и стоимость для одного человека. Поход имеет свой маршрут, по которому идет группа.

В предметной области есть города, которые принадлежат странам. Впорос может относиться к какому-то городу, также он должен быть привязан к какому-то пользователю системы.

У одного вопроса может быть много ответов.

Каждый ответ должен обязательно иметь пользователя, который его написал. У города могут быть места, которые расположены в этом городе.

Ответ обязательно должен быть связан с каким-то вопросом.

Все вопросы могу иметь список голосов. У каждого голоса должен быть пользователь, который его сделал.

Все ответы могу иметь список голосов. У каждого голоса за ответ должен быть пользователь, который сделал этот голос. На основании длины этого списка формируется рейтинг вопроса или ответа. Ответы или вопросы с большим рейтингом имеют больший приоритет.

После исследования предметной области можно составить диаграмму взаимодействия (см. рис.1.1).

Из данной функциональной структуры можно сделать вывод, что в системе существует турист, который путешествует по миру. Также он может ходить в походы. Также у него могу быть вопросы. Вместе она представляют собой подсистему, которая взаимодействует между собой.

Необходимо выделить участников системы, которые принимают решения и их информационные нужды. Гости системы и зарегистрированные пользователи.

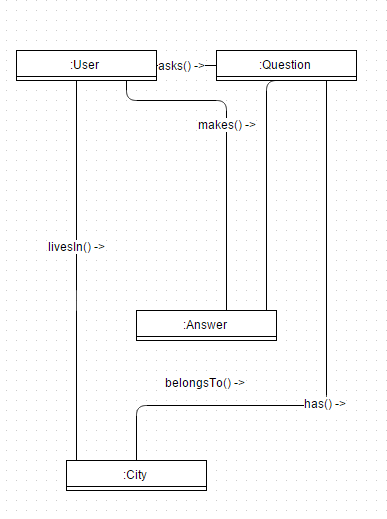


Рисунок 1.1 Функциональная структура исследуемой системы

Информационные нужды руководителя и администраторов:

* знать список вопросов;
* иметь информацию о городах;
* иметь информацию о местах;
* иметь возможность просмотреть список ответов для вопроса;
* знать дату вопроса и дату ответа;
* знать рейтинг вопроса и ответа;
* знать список вопросов для одного пользователя;

Далее будут рассмотрены объекты, которые должны храниться в базе данных и конкретная информация о них.

Клиент:

* фамилия;
* имя;
* пол;
* дата рождения (число.месяц.год);
* инфрмация о пользователе;
* фото;
* страна.

Город:

* название;
* фото;
* страна.

Страна:

* название;
* города.

Вопрос:

* заголов;
* текст;
* дата;
* автор.

После анализа предметной области были выявлены следующие ограничения целостности:

* клиент может быть идентифицирован по логину;
* поход может быть идентифицирован по дате начала и маршруту;
* вопрос и ответ имеют ссылку на клиента;
* ответ может иметь неограниченног количество голосов;
* вопрос может иметь неограниченное количество голосов.

Необходимо рассмотреть процессы потребления информации и существующие в системе документы. Информация поступает от клиента к программной системе в виде текста из формы с заполненной информацией об участнике вопросе[1]. Потом клиент может просматривать список все вопросов, своих вопросов, ответов на эти вопросы. Он может отвечать на другие вопросы. Таким образом пользователи обмениваются информацией между собой. Также они могут зайти на страницу друг другая, чтобы посмотреть список вопросов и информацию о другом пользователе.

2 ПОСТАНОВКА ЗАДАЧИ

В курсовом проекте необходимо спроектировать и реализовать программную систему. В этом разделе перечислены задачи, которые должна решать информационная система.

Система должна отображать данные страны, города, вопросы, ответы, клиентов и информации о них.

Система должна обеспечивать добавление нового вопроса, ответа, пользователя, страны.

Система должна предоставлять возможность просмотреть список вопросов во всем приложении, список вопросов по городам, список вопросов по одному пользователю или городу.

Система должна отображать список ответов к вопросу.

Система дожна предоставлять возможность поиска по ключевому слову среди списка всех вопросов.

Так же она должна предоставлять возможность отслеживать рейтинг вопросов и ответов.

В системе должна быть предусмотрена возможность зарегистрироваться.

Зарегистрированные пользователи должна иметь возможность войти в систему под своим логином.

После этого они должны иметь возможность проголосовать за какой-то вопрос или ответ.

Еще пользователи, которые вошли в систему, имеют возможность ответить на вопрос, на который они знают ответ.

Система должна отображать список ответов к вопросу.

Система дожна предоставлять возможность поиска по ключевому слову среди списка всех вопросов.

Зарегистрированные пользователи должна иметь возможность войти в систему под своим логином.

Система дожна предоставлять возможность поиска по ключевому слову среди списка всех вопросов.

В системе должна быть предусмотрена возможность зарегистрироваться.

Система должна предоставлять возможность просмотреть список вопросов во всем приложении, список вопросов по городам, список вопросов по одному пользователю или городу.

Система должна хранить информацию о пользователе. Ниже рассмотрен пример такой информации (см. таб. 2.1).

Таблица 2.1 – Пример данных о пользователе

|  |  |
| --- | --- |
| Имя | Иван |
| Фамилия | Иванов |
| Пол | М |
| Страна | Украина |
| Города | Харьков, Киев |
| Логин | Ivani |
| Пароль | 23423sdfk312 |
| Фото | http://localhost:8080/img/1.jpg |
| Обо мне | Родился в городе Харьков, проживаю сейчас в городе Киев. Знаю много об этих городах. Могу рассказать много интересного. Ответить на ваши вопросы касательно этих двух городов и Украины. |

Также система должна предоставлять возможность отсортировать ответы или вопросы по их рейтингу.

Система должна предоставлять возможность поиска по ключевому слову среди списка всех вопросов. Зарегистрированные пользователи должна иметь возможность войти в систему под своим логином. Система должна предоставлять возможность поиска по ключевому слову среди списка всех вопросов.

Требованием также есть создание трехуровневой масштабируемой системы.

В этом разделе были представлены задачи, которые требуют реализации. Также была выбрана задача автоматизации – регистрация пользователя по электронному заявлению.

3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

3.1 UML – моделирование

На основе поставленных задач была построена диаграмма прецедентов (см. рис.3.1).

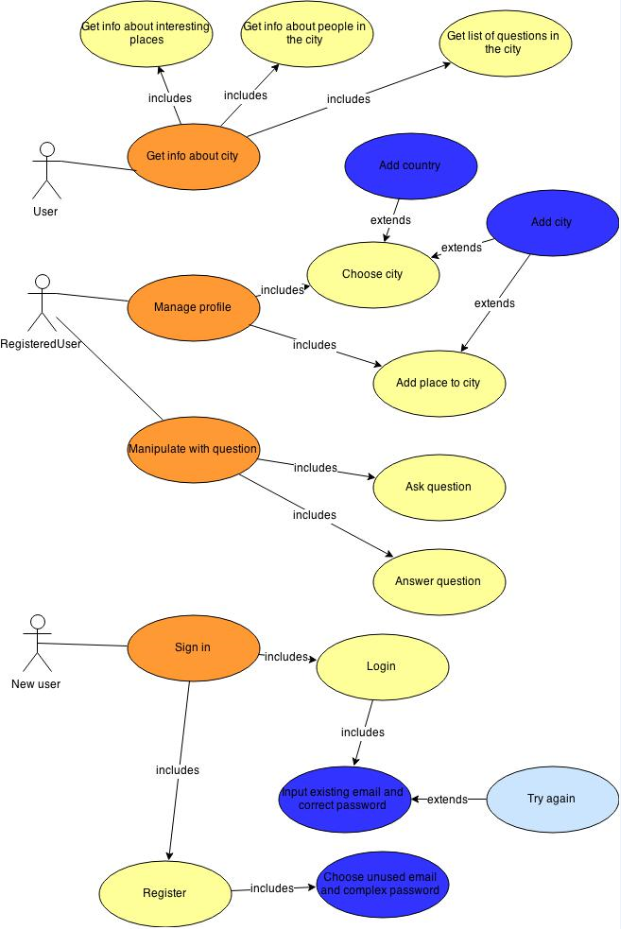


Рисунок 3.1 – Диаграмма прецедентов

Данная диаграмма отображает поставленные задачи. Как видно, в системе пользователи разделяются, для системы существует вид пользователей. Из диаграммы мы узнаем, что пользователь может просмотреть информацию о сущности (следует отметить, что у этого элемента есть наследники, которые уточняют его и говорят о возможности посмотреть про конкретную сущность, например, про инструкторов и т.д.)[6]. С каждым элементом связан прецедент, который показывает возможность выполнить часто используемый запрос, подготовить заявление в контрольно-спасательную службу из окна просмотра походов, просмотреть список наиболее популярных маршрутов.

Кроме просмотра информации также существуют прецеденты, которые указывают на возможность удаления или редактирования информации. По аналогии с просмотром информации существуют наследники, которые конкретизируют сущность, над которой выполняется операция. Также существует элемент Use Case, который указывает на возможность добавления клиента с помощью заполненного заявления клиента.

Кроме перечисленных функций также следует отметить элемент «Выполнить произвольный запрос», который указывает на возможность вручную ввести интересующий запрос и получить по нему результат. Этот элемент не связан с другими, потому что его функциональность не зависит от других прецедентов.

Есть элемент «Получить отчет о прибыльности фирмы», который подразумевает собой возможность просмотреть или сохранить документ, который будет наполнен информацией о походах и данными о том, сколько какой поход принес денег и сколько было потрачено на дополнительные расходы и зарплату руководителю похода.

Также из этой диаграммы видно, что пользователь имеет возможность задавать вопросы в программе. И отвечать на вопросы, которые задают другие пользователи.

Еще зарегистрированный пользователь может менять рейтинг других вопросов или ответов.

Любой пользователь имеет возможность просмотреть рейтинг уже существующих вопросов и ответов.

Для облегчения понимания работы программы было решено построить Activity diagram (см. рис.3.2).

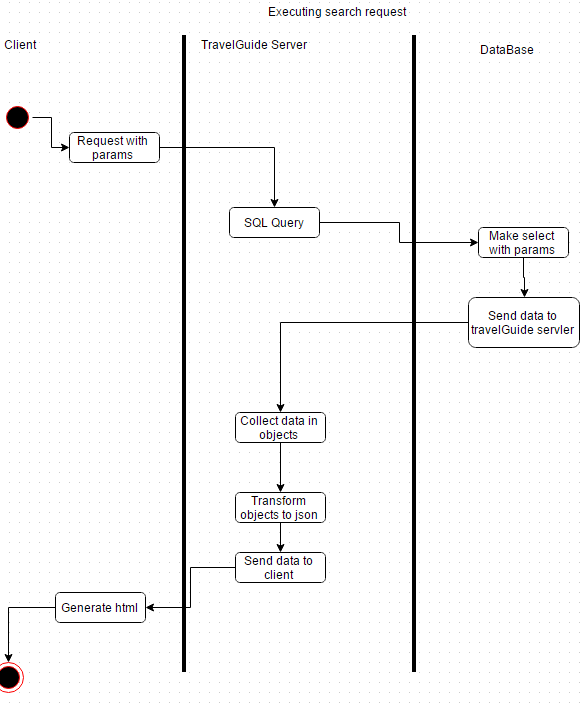


Рисунок 3.2 – Activity diagram

На этой диаграмме мы видим последовательно выполняемую задачу выборки данных из базы данных. На первом этапе необходимо выбрать данные для обработки. Это может быть название объекта по которому будет производится выборка. Файл представляет собой, в которой есть два столбца с названием данных и значением парами. Например, «Фамилия» – «Иванов». После эта информация анализируется и на ее основе происходит добавление нового участника.

Также было решено построить диаграмму состояний, чтобы было легче понять, как происходит работа подключений (см. рис.3.3).

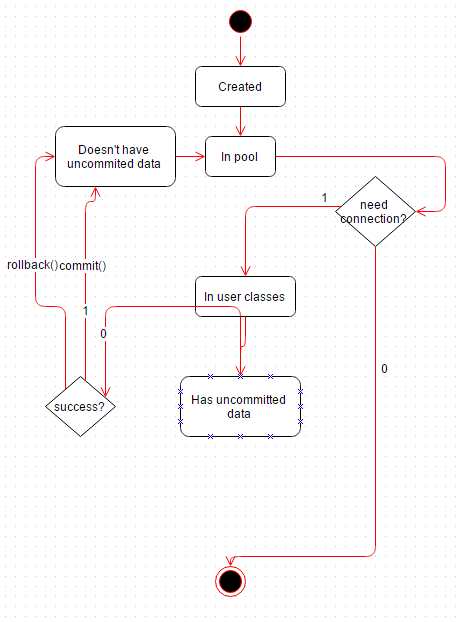


Рисунок 3.3 – ER-диаграмма

Выше представлена полная диаграмма с приведением всех атрибутов, которые будут у сущностей.

На данной диаграмме можно наблюдать связь многие ко многим между походом и клиентом, которая реализована с помощью дополнительной сущности «заявка». Ключом сущности «Клиент» есть № паспорта клиента, «Поход» - № похода, «заявка» - № заявки, кроме этого заявка содержит первичные ключи похода и клиента для связки.

Существует связь между инструктором и походом один ко многим, которая реализуется с помощью внесения в поход номера паспорта инструктора.

Также на диаграмме присутствует связь один ко многим (поход и маршрут с помощью поля «№ маршрута» у похода) и многие ко многим (маршрут и место через дополнительную сущность стоянка маршрута).

3.2 Построение UML диаграмм реализаций

Из концептуального анализа предметной области получили набор атрибутов, который следует нормализировать.

На рисунке 3.4 изображена диаграмма классов [4]. Эта диаграмма отображает отношение между классами программы. Также там перечислены основные методы и атрибуты классов. Показаны такие связи между классами как наследования, зависимость, агрегация и еще. Таблица в целях наглядности разделена на две, хотя является целой. Все атрибуты неделимы, нет таких атрибутов, как, например, ФИО, существует три поля, хранящие имя, фамилию и отчество. Также нет повторяющихся строк, существуют уникальные ключи. В качестве уникального ключа выступает пара. По этим двум значениям можно определить значения всех остальных атрибутов.

Таблица в целях наглядности разделена на две, хотя является целой. Все атрибуты неделимы, нет таких атрибутов, как, например, ФИО, существует три поля, хранящие имя, фамилию и отчество. Также нет повторяющихся строк, существуют уникальные ключи. В качестве уникального ключа выступает пара. По этим двум значениям можно определить значения всех остальных атрибутов.

На этой диаграмме отображены основные сущности в программе, которые играют самую важную роль.

Эта диаграмма имеет очень большую роль так как она отображает большую часть дизайна системы. По ней можно понять основные взаимодействия между классами программы.

Еще можно узнать, как классы относятся друг к другу. Между некоторыми классами может отношение наследования. Таким образом классы перенимают какую-то базовую функциональность и дополняют ее своими функциями.

Неотъемлемой частью является реализация интерфейсов. Таким образом часть программы может инкапсулировать в себе какую-то реализацию программы, чтобы остальные классы не были зависимы от реализации.

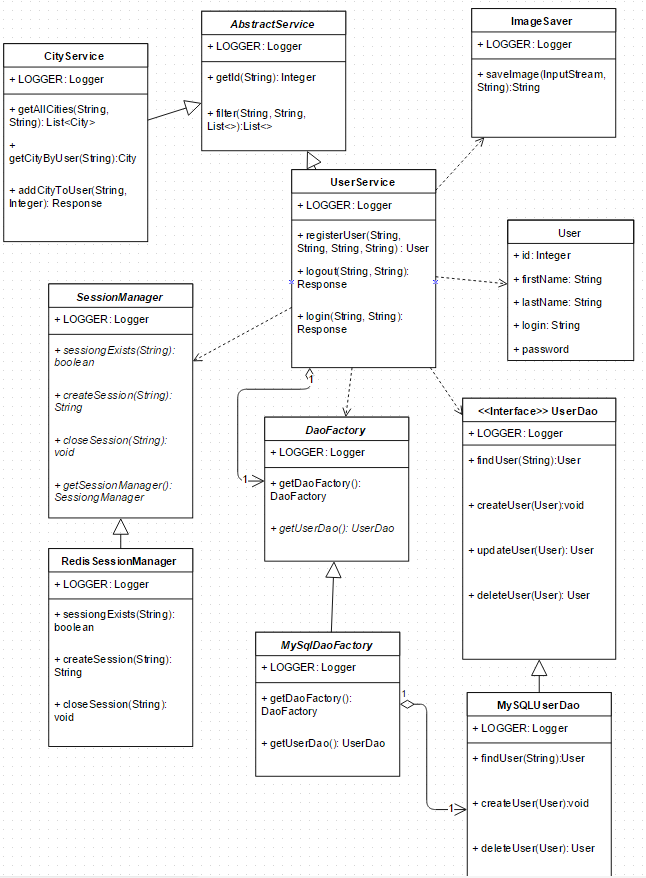


Рисунок 3.4 – Диаграмма классов

Данная диаграмма отображает отношения между классами. Как можем видеть там изображено несколько классов программы и несколько интерфейсов.

Было решено разработать диаграмму объектов, которую можно условно назвать срезом диаграммы классов. Она показывает значение переменных классов в определенный момент времени. Данное отношение не находится во второй нормальной форме, потому что есть неключевые атрибуты, которые зависят от частей составного ключа. Ниже изображена диаграмма объектов (см. рис. 3.5).

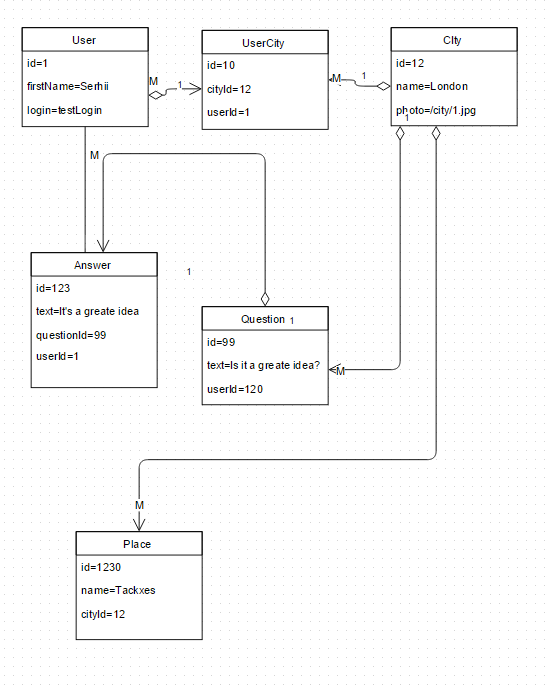


Рисунок 3.5 – Отношения во второй нормальной форме

Мы получили диаграмму объектов, которая показывает состояние программы в определенный момент времени.

На диаграмме объектов расположены главные объекты программной системы, которые могут взаимодействовать между собой.

Далее будет рассмотрена диаграмма последовательности (см. рис. 3.6).

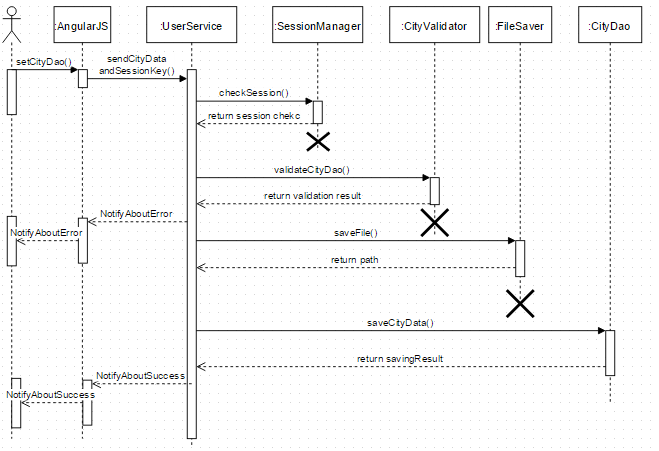


Рисунок 3.6 – Отношения в третьей нормальной форме

Таким образом из данной диаграммы можно наблюдать взаимодействие между пользователем и программной системой. Сначала он передает информацию о городе на клиентский сайт. Потом он передает эту информацию на сервер с бизнес логикой, который производит запрос в базу данных, откуда выбираются данные и отправляются обратно пользователю.

Также можно заметить, что на каждом этапе пользователю можно отправить сообщение об ошибке.

Большинство объектов на этой диаграмме умирают после того, как произойдет взаимодействие с другой программой.

Таким образом было произведено проектирование программной системы.

4 ОПИСАНИЕ ПРОГРАММЫ

4.1 Общие сведенья

Разрабатываемый программный продукт создавался в интегрированной среде разработки IntelliJ IDEA. В качестве СУБД была выбрана MySQL. Ее основные преимущества:

* лучшая скорость обработки данных на объеме до 500000 записей;
* наличие бесплатных открытых лицензий;
* простота использования, поддержка многих платформ;
* поддержка нескольких одновременных запросов;
* наличие большого количества встроенных функций.

Также эта СУБД поддерживается IDEA, благодаря чему разработка стала более легкой и автоматизированной [7].

Требования к оборудованию:

* процессор с тактовой частотой 1 ГГц или выше;
* ОЗУ объемом 512 мегабайт или больше;
* 2 мегабайта свободно пространства на жестком диске;
* установленная Java 1.7.

4.2 Вызов и загрузка

Для того чтобы запустить программу необходимо открыть папку с программой и запустить программу maven из командной строки параметрами tomcat7:run. После этого откроется главное окно программы. Далее нужно перейти в папке с клиентским приложением. И ввести команду npm install, npm start.

Программная система сразу готова к использованию. Пользователю не нужно создавать псевдоним базы данных. Система сама подключается к ней тогда, когда ей это необходимо. При этом информационная система подразумевает, что база данных находится на localhost и порту 3336. Подключение осуществляется с помощью имени пользователя и пароля. Эта процедура выполняется неявно при необходимости получить какие-то данные из базы.

4.3 Назначение и логическая структура

В программной системе были реализованы все функции, перечисленные в постановке задачи. Ниже приведен этот список и краткое описание модулей, которые отвечают за выполнение функции.

Отображение данных о клиентах, вопросах и ответах, походах и маршрутах реализовано в клиентском приложении.

Изменение и удаление информации о сущностях реализовано в соответствующих формах, таких как EditClientForm, EditRoutForm и т. д.

Для удаления данных предусмотрена кнопка в главном окне программы. Она удаляет информацию из базы данных о выбранной в таблице сущности.

Для поиска и фильтрации информации возле каждой таблицы отображения информации о сущностях расположены поля для ввода информации для поиска и фильтрации и соответствующие кнопки.

Функции узнать список туристов, которые сейчас находятся в городе, список наиболее популярных маршрутов, список походов, которые сейчас идут реализованы в модуле MainForm.

Возможность формирования собственного запроса реализована в модуле QueryEditorForm.

Система также может сформировать заявку в контрольно-спасательную службу. С точки зрения программной реализации эта заявка представляет собой отчет, котором перечислена необходимая информация. Поэтому был создан отчет ReportApp.rdlc и размещен на форме CSSapplicationForm.

В программной система реализована возможность живого поиска по вопросам. Эта функциональность реализована на главной странице клиентского приложения. На этой странице над списком всех фильмов расположена элемент html, который отвечает за ввод данных. После этого можно сфокусироваться на этом поле ввода и начать вводить часть заголовка вопроса. Через некую долю секунды список автоматически отфильтрует все вопросы по заданному полю ввода.

Так же есть функция входа в систему. Она реализована на верхнем навигационном поле. После нажатия на кнопку появится диалоговое окно для входа в систему.

Реализована функция голосования за вопросы и также. По умолчанию все ответы отсортированы в порядке уменьшения рейтинга.

4.4 Описание физической модели базы данных

В качестве СУБД была выбрана MySQL, поэтому описание физической модели представлено в виде скриншотов из этой программы с описанием.

Структура таблицы «User» приведена на рисунке 4.1.

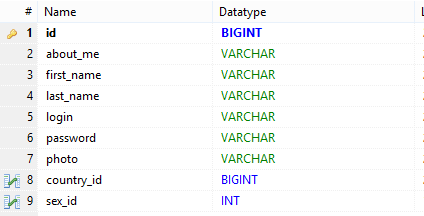


Рисунок 4.1 – Структура таблицы «User»

На приведенном рисунке перечислены поля таблицы «Клиент». В качестве первичного ключа обозначено поле «ID», на которое наложено ограничение не пустое и уникальное. Внешние ключи: страна, пол.

Структура таблицы «City» приведена на рисунке 4.2.

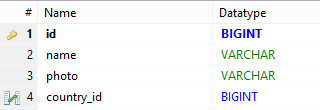


Рисунок 4.2 – Структура таблицы «City»

На приведенном рисунке перечислены поля из таблицы «City». В качестве первичного ключа обозначено поле «id», аналогично таблице «Клиент». Внешний ключ ссылается на таблицу страна.

Таблица «Question» приведена на рисунке 4.3.

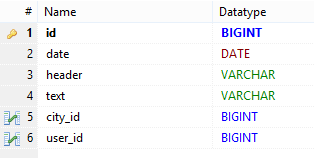


Рисунок 4.3 – Структура таблицы «Question»

На рисунке выше приведена структура таблицы «Question». В качестве первичного ключа выбрано поле «id», в котором помимо обычных ограничений также отмечено «AI», что означает, что поле автоматически растет. Это позволяет сделать целочисленный тип поля. Таблица ссылается на город и пользователя. Других ограничений на таблицу не наложено.

На рисунке 4.4 приведена структура таблицы «Answer».

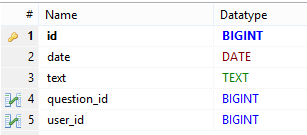


Рисунок 4.4 – Структура таблицы «Answer»

На приведенном рисунке описана структура таблицы «Answer». В качестве первичного ключа выбрано поле «id» с опцией автоматического инкрементирования. В этой таблице есть внешние ключи, которые ссылаются на таблицы «Question» и «User». У внешних ключей выбраны опции: при удалении инструктора или маршрута выставить соответствующие поле в NULL, при обновлении – изменить.

На рисунке 4.5 приведена схема таблицы «question\_vote».

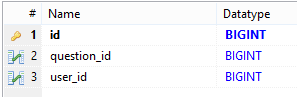


Рисунок 4.5 – Структура таблицы «question\_vote»

В качестве первичного ключа выбрано поле «id». Внешние ключи ссылаются на таблицу вопросов и пользователей. С внешними ключами выбраны опции каскадирования. При удалении клиента или похода, удаляется и сама заявка.

По данным в этой таблице осуществляется подсчет рейтинга для ответа.

Также потом по этой таблице можно будет узнать список пользователей, которые проголосовали за данный вопрос

На рисунке 4.6 приведена схема таблицы «Answer\_Vote».

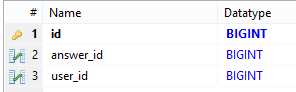


Рисунок 4.6 – Структура таблицы «Answer\_vote»

Дополнительные опции этой таблицы схожи с таблицей «Question\_vote».

На рисунке 4.7 представлена структура таблицы «Country».

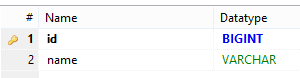


Рисунок 4.7 – Структура таблицы «Country»

Эта таблица не имеет никаких дополнительных ограничений.

Эти таблицы вместе объединены в базу данных htmdb, которая расположена в MySQL. Эти таблицы взаимодействуют между собой. Сервер с бизнес логикой обращается к ним и посредством SQL может получить доступ к каким-либо данным.

Также есть прослойка между логикой приложения и базой данных в виде Hibernate, который предоставляет более удобный интерфейс. Также оно предоставляет более легкий способ организации транзакций.

Эта библиотека предоставляет удобный способ отображения объектом предметной области на таблицы в базе данных.

Эта таблица имеет первичный ключ «id». Также она ссылается на таблицы город, к которому принадлежит это место и пользователю, который добавил это место.

4.5 Описание программной реализации

В этом разделе будут приведены экранные формы, которые реализуют перечисленные ранее функции.

Экранные формы будут представлены из браузера с открытым клиентским приложением.

Отображение списка всех вопросов представлено на рисунке 4.8.

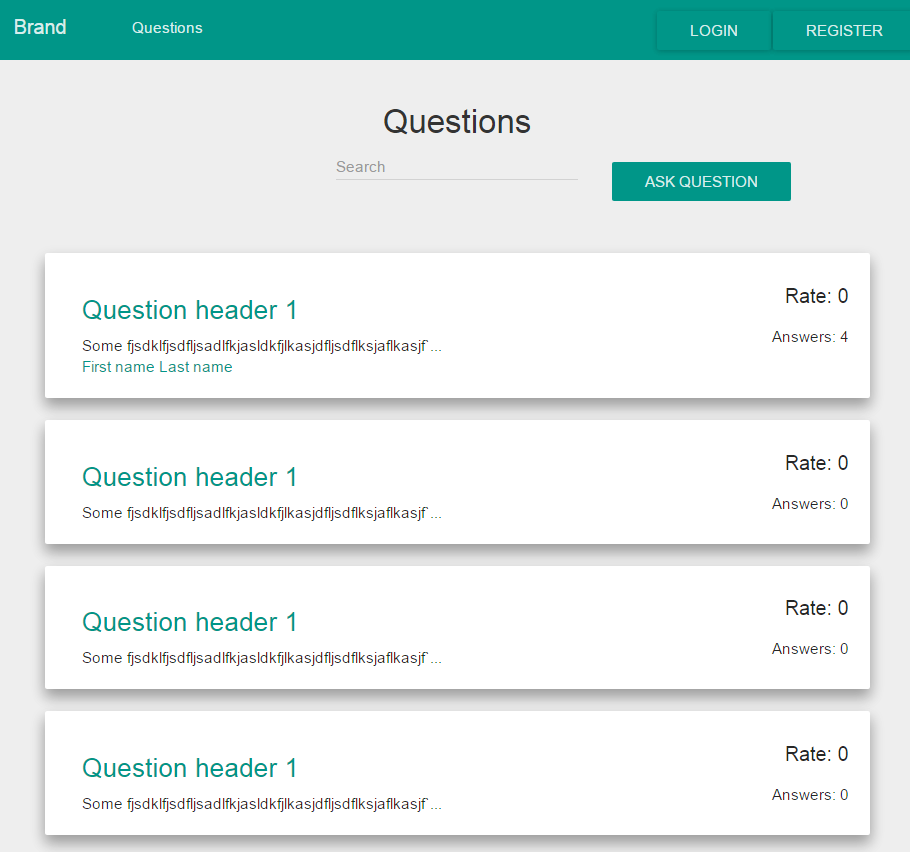


Рисунок 4.8 – Отображение списка вопросов

На этой странице существует возможность делать поиск по ключевому слову. На ней еще можно менять рейтинг вопросов. И просмотреть первые 100 символов вопроса. Для каждого вопроса можно просмотреть количество ответов, которые были добавлены для каждого вопроса.

Аналогично реализовано отображения информации об инструкторах, походах и маршрутах во вкладках главной формы.

Страница просмотра одного просмотра представлена на рисунке 4.9.

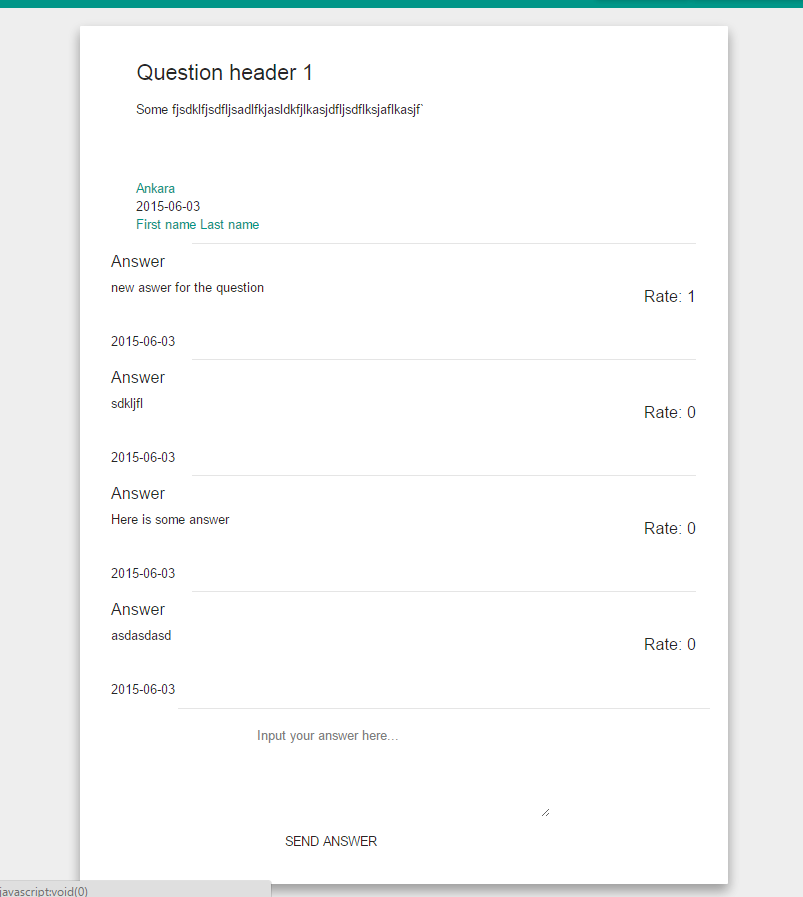


Рисунок 4.9 – Страница просмотра вопроса

На этой странице можно просмотреть список ответов к одному вопросу. Можно просмотреть рейтинг каждого вопроса. Также, если пользователь вошел в систему, то он может менять рейтинг вопроса добавляя или отнимая от него свой голос.

Следует обратить внимание на страницу просмотра пользователя, которая представлена на рисунке 4.10.

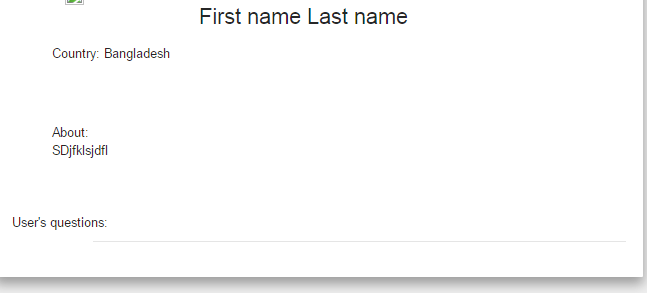


Рисунок 4.10 – Страница пользователя

На этой форме мы может просмотреть список вопросов, которые задал конкретный пользователь. Можно просмотреть рейтинг конкретного вопроса или изменить его.

В системе присутствует авторизация, поэтому есть формы входа и логина. Форма логина представлена на рисунке 4.11.

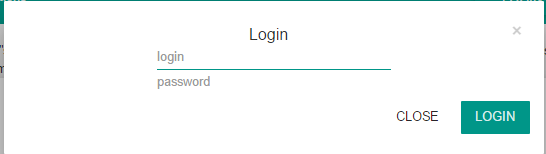


Рисунок 4.11 – Форма входа в систему

На приведенной выше форме пользователю необходимо ввести логин и пароль для входа в систему.

После этого диалог исчезает и либо появляется ошибка, либо пользователь успешно входит в систему. Ошибка возникает если пользователь неправильно ввел логин или пароль.

Также можно нажать на кнопку «Close», которая закроет форму без попытки логина.

Только что были рассмотрены некоторые страницы, которые есть в клиентском приложении.

4.6 Описание выбранных технологий

Далее будут рассмотрены технологии и фреймворки, которые использовались в этом приложении.

Для начала рассмотри клиентское приложение.

Во время написания клиентского приложения были использованы AngularJS, Bootstrap 3.0, Bootstrap Material Design, JQuery.

AngularJS - JavaScript-фреймворк с открытым исходным кодом. Предназначен для разработки одностраничных приложений. Его цель — расширение браузерных приложений на основе MVC шаблона, а также упрощение тестирования и разработки.

Фреймворк работает с HTML, содержащим дополнительные пользовательские атрибуты, которые описываются директивами, и связывает ввод или вывод области страницы с моделью, представляющей собой обычные переменные JavaScript. Значения этих переменных задаются вручную или извлекаются из статических или динамических JSON-данных.

Двустороннее связывание данных в AngularJS является наиболее примечательной особенностью и уменьшает количество кода, освобождая сервер от работы с шаблонами. Вместо этого, шаблоны отображаются как обычный HTML, наполненный данными, содержащимися в области видимости, определенной в модели. Сервис $scope в Angular следит за изменениями в модели и изменяет раздел HTML-выражения в представлении через контроллер. Кроме того, любые изменения в представлении отражаются в модели. Это позволяет обойти необходимость манипулирования DOM и облегчает инициализацию и прототипирование веб-приложений [2].

AngularJS использовался для упрощения взаимодействия между компонентами клиентского приложения.

Twitter Bootstrap использовался для создания приятного интерфейса.

свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов, включая JavaScript расширения.

Bootstrap использует самые современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров.

Экономия времени — Bootstrap позволяет сэкономить время и усилия, используя шаблоны дизайна и классы, и сконцентрироваться на других разработках.

Высокая скорость — динамичные макеты Bootstrap масштабируются на разные устройства и разрешения экрана без каких-либо изменений в разметке.

Гармоничный дизайн — все компоненты платформы Bootstrap используют единый стиль и шаблоны с помощью центральной библиотеки. Дизайн и макеты веб-страниц согласуются друг с другом.

Простота в использовании — платформа проста в использовании, пользователь с базовыми знаниями HTML и CSS может начать разработку с Twitter Bootstrap;

Совместимость с браузерами — Twitter Bootstrap совместим с Mozilla Firefox, Google Chrome, Safari, Internet Explorer и Opera.

Bootstrap Material Design использовался для того, чтобы применить стиль материальности к интерфейсу. Это надстройка над Twitter Bootstarp, которая дополняет его стили по умолчанию.

Для построения приложения бизнес логики использовался фреймворк Spring, Spring MVC, Hibernate.

Несмотря на то, что Spring Framework не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring Framework предоставляет большую свободу Java-разработчикам в проектировании, кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring Framework применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

КОНКУРЕНТЫ

В данном разделе будут рассмотрены основные конкуренты разрабатываемого приложения.

Аналогом данного приложения есть сервис StackOverflow.

Stack Overflow — популярная система вопросов и ответов о программировании, разработанная Джоэлем Спольски и Джеффом Этвудом (англ.) в 2008 году. Является частью Stack Exchange Network. Как и в других системах подобного рода, Stack Overflow предоставляет возможность оценивать вопросы и ответы, что поднимает или понижает репутацию зарегистрированных пользователей (вариант игрофикации).

На июль 2014 года в базе данных веб-сайта хранилось почти 7,7 миллионов вопросов. В восьмерку наиболее популярных тем, согласно указываемым тегам, на Stack Overflow входят Java, C#, JavaScript, PHP, Android, jQuery, Python и HTML.

Но этот сервия предназначен для вопросов по программированию. Разрабатываемая система же является аналогом этого сервися, но может быть использована туристам и путешествинниками, которые хотят узнать какую-то информацию о городе, в который они направляются.

Более близким кокурентом является сервис Tripster.

Они предлогают необычные экскурсии от местных жителей – это увлекательное погружение в город, его историю, культуру и секреты для тех, кто ищет настоящий дух места.

На сайте у них можно выбрать гида, или экскурсию. Также можно задать вопрос по городу.

Отличием от конкурентов можно назвать наличие onepageapplication.

ВЫВОДЫ

В данном курсовом проекте была спроектирована и создана база данных и информационная система для удобного взаимодействия пользователей. Также была освоена работа с выбранной СУБД - MySQL. Были освоены такие технологии как Spring Framework, Spring MVC, Hibernate, AngularJS, Bootstrap Material Design.

Был произведен концептуальный анализ предметной области, который использовался далее для создания базы данных и информационной системы.

База данных была разработана для туристов и хранит в себе информацию об пользователях, вопросах и ответах, рейтингах, через которые проходят маршруты. В ней реализованы связи один ко многим и многие ко многим через вспомогательные сущности. К примеру, связь между клиентом и походом осуществляется через сущность «заявка», которая хранит в себе внешние ключи на записи из таблиц «Клиент» и «Поход».

Во время проектирования базы данных были также изучены первые три нормальные формы и применены на практике. База данных приведена к третьей нормальной форме, что является компромиссом между быстродействием и качеством структуры базы данных.

Во время выбора СУБД были изучены основные преимущества самых распространённых СУБД, таких как Oracle, MySQL, Microsoft SQL Server. Была выбрана вторая.

Во время создания программы было изучено взаимодействие приложения на технологии Java и СУБД MySQL. Изучены такие элементы, как DAOFactory, Resultset и другие, которые помогали использовать информацию из базы данных, изменять ее и отображать на экране.

Были также приобретены практические навыки использования SQL при доступе к базе данных. Написаны структурно сложные запросы, которые получают структурированную информацию из базы данных.

Были выполнены все требования, которые были перечислены в разделе постановки задачи.

ПЕРЕЧЕНЬ ССЫЛОК

1. http://www.outdoorukraine.com/
2. Дейт К.Дж. Введение в системы баз данных [Текст] / К.Дж. Дейт. - 8-е изд. – М.:Издательский дом „Вильямс”, 2005.- 1328 с.
3. Мартин Р. Чистый код. Создание, анализ и рефакторинг [Текст]/ Р. Мартин; Питер. - 464 с.
4. Грабер М. SQL. [Текст] / М.Грабер.- К.: Изд-во “ЛОРИ”,2003.–644 с.
5. Гарсия-Молина Г. Системы баз данных. Полный курс. [Текст]: Пер. с англ./ Г.Гарсия-Молина, Дж.Ульман, Дж.Уидом. – М.: Издательский дом „Вильямс”, 2003. – 1088 с.
6. Кренке Д. Теория и практика построения баз данных. [Текст] / Д.Кренке - 8-е изд. – СПб.: Питер, 2003. – 800 с.
7. <http://strong-planet.ru/database.html>
8. Герберт Шилдт "C# 4.0. Полное руководство".
9. <http://msdn.microsoft.com>
10. http://stackoverflow.com/