**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

Факультет СПО (название факультета)

Направление подготовки (специальность) 09.02.07 Информационные системы и программирование

**О Т Ч Е Т**

**о производственной практике (по профилю специальности)**

**по ПМ.11 «Разработка, администрирование и защита баз данных»**

(наименование практики)

Тема задания: Разработка АИС для виртуальных петиций

Обучающийся \_\_\_Шульчин Р.Д. Группа \_\_\_\_Y2433\_\_\_\_\_\_\_

(Фамилия И.О.) (номер группы)

Руководитель практики от организации: Слабкий Андрей Александрович, ведущий программист отдела сетевых технологий филиала акционерного общества «Концерн радиостроения «Вега» в г. Санкт-Петербурге

(Фамилия И.О., должность и место работы)

Руководитель практики от факультета: Говорова М.М., преподаватель  
Ефимова Т.Н., преподаватель

(Фамилия И.О., должность и место работы)

Ответственный за практику от университета: Королёв В.В., заместитель директора факультета СПО

(Фамилия И.О., должность)

Практика пройдена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии\_\_\_\_\_\_\_\_\_\_\_( )

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Говорова М.М.)

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Ефимова Т.Н.)

(подпись)

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2020

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ 2

ВВЕДЕНИЕ 3

проектирование и программная реализация системы 5

ЗАКЛЮЧЕНИЕ 30

список источников 31

ПРИЛОЖЕНИЕ А 32

ПРИЛОЖЕНИЕ Б 42

ВВЕДЕНИЕ

Цель производственной практики – углубление знаний и практических умений и получение практического опыта по виду деятельности «Разработка, администрирование и защита баз данных» и овладение следующими профессиональными компетенциями:

* осуществлять сбор, обработку и анализ информации для проектирования баз данных;
* проектировать базу данных на основе анализа предметной области;
* разрабатывать объекты базы данных в соответствии с результатами анализа предметной области;
* реализовывать базу данных в конкретной системе управления базами данных;
* администрировать базу данных;
* защищать информацию в базе данных с использованием технологии защиты информации.

Место прохождения практики – отдел сетевых технологий филиала акционерного общества «Концерн радиостроения «Вега» в г. Санкт-Петербурге.

Задачи по разработке системы:

* проанализировать индивидуальное задание;
* определить требования к разработке;
* смоделировать поведение системы;
* сформировать техническое задание;
* реализовать серверную и клиентскую часть системы;
* подготовить отчетную документацию.

Практическая значимость разрабатываемой автоматизированной информационной системы (далее – АИС) – своевременное обеспечение пользователей соответствующей информацией, возможность ее хранения, поиска и обработки.

Отчет по практике содержит информацию о методологиях проектирования системы, архитектуре системы, обзор программных средств для реализации системы, модели функционального поведения системы и базы данных, программную реализацию системы.

1. проектирование и программная реализация системы
   1. Методологии проектирования системы

Принципы, законы и методы имеются в любой деятельности. Есть они и в проектировании. Люди всегда пытаются отыскать их, чтобы не только облегчить «муки творчества», но и создать универсальные «решатели задач». Однако основные открытия были сделаны лишь во второй половине 20 века, когда было провозглашено рождение «общества потребления», цель которого — максимально полное удовлетворение потребностей человека, что возможно только при высокой производительности творческого труда.

Системы, которые осуществляют хранение и обработку информации называют информационно-вычислительными системами. В информационную систему данные поступают от источника информации. Эти данные отправляются на хранение либо претерпевают в системе некоторую обработку и затем передаются потребителю.

Проектирование представляет собой последовательность выполнения взаимообусловленных действий — процедур. В свою очередь, процедуры подразумевают использование определенных методов, основанных на тех или иных законах природы и общества.

На этапе проектирования формируются модели данных. Проектировщики в качестве исходной информации получают результаты анализа требований к ИС. Построение логической и физической моделей данных является основной частью проектирования базы данных. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных. Параллельно с проектированием схемы базы данных выполняется проектирование процессов, чтобы получить спецификации (описания) всех модулей ИС. При проектировании модулей определяют интерфейсы программ: разметку меню, вид окон, горячие клавиши и связанные с ними вызовы.

Для построения моделей информационных систем используется методология UML (Unified Modelling Language). В методологии UML функции системы представляются на диаграмме прецедентов (диаграммы вариантов использования, Use Case диаграммы), где представляются акторы и действия, которые они могут выполнять в системе.

Для построения модели поведения системы используется методология DFD (Data Flow Diagram). Целью методологии DFD является построение модели системы в виде диаграммы потоков данных. При создании DFD используются понятия:

потоки данных, моделирующие передачу информации и изображающиеся в виде именованных стрелок;

процессы преобразования входных потоков данных в выходные. Процессы изображаются в виде скругленных прямоугольников, а внутри помещается название процесса в виде глагола в неопределенной форме.

Модель «сущность – связь» (Entity – Relationship, ER) позволяет описать логическую структуру базы данных. Известны несколько нотаций ER-моделей, наиболее популярной является IDEF1.

Сущность – это абстрактный объект, имеющий в конкретном контексте независимое существование, а связь – это ассоциация сущностей. Атрибут – свойство сущности или связи.

* 1. Архитектура системы

Архитектура программной системы охватывает не только ее структурные и поведенческие аспекты, но и правила ее использования и интеграции с другими системами, функциональность, производительность, гибкость, надежность, возможность повторного применения, полноту, экономические и технологические ограничения, а также вопрос пользовательского интерфейса.

По мере развития программных систем все большее значение приобретает их интеграция друг с другом с целью построения единого информационного пространства предприятия. Как можно видеть из вышеприведенных определений интеграция является важнейшим элементом архитектуры.

Для того чтобы построить правильную и надежную архитектуру и грамотно спроектировать интеграцию программных систем необходимо четко следовать современным стандартам в этих областях. Без этого велика вероятность создать архитектуру, которая неспособна развиваться и удовлетворять растущим потребностям пользователей ИТ. В качестве законодателей стандартов в этой области выступают такие международные организации как SEI (Software Engineering Institute), WWW (консорциум World Wide Web), OMG (Object Management Group), организация разработчиков Java – JCP (Java Community Process), IEEE (Institute of Electrical and Electronics Engineers) и другие.

Рассмотрим классификацию программных систем по их архитектуре:

* + - Централизованная архитектура;
    - Архитектура "файл-сервер";
    - Двухзвенная архитектура "клиент-сервер";
    - Многозвенная архитектура "клиент-сервер";

1. Централизованная архитектура вычислительных систем.

Была распространена в 70-х и 80-х годах и реализовывалась на базе мейнфреймов (например, IBM-360/370 или их отечественных аналогов серии ЕС ЭВМ), либо на базе мини-ЭВМ (например, PDP-11 или их отечественного аналога СМ-4) [4]. Характерная особенность такой архитектуры – полная "неинтеллектуальность" терминалов. Их работой управляет хост-ЭВМ.

Достоинства такой архитектуры:

* пользователи совместно используют дорогие ресурсы ЭВМ и дорогие периферийные устройства;
* централизация ресурсов и оборудования облегчает обслуживание и эксплуатацию вычислительной системы;
* отсутствует необходимость администрирования рабочих мест пользователей;

Главным недостатком для пользователя является то, что он полностью зависит от администратора хост-ЭВМ. Пользователь не может настроить рабочую среду под свои потребности – все используемое программное обеспечение является коллективным.

Использование такой архитектуры является оправданным, если хост-ЭВМ очень дорогая, например, супер-ЭВМ.

Центральная ЭВМ должна иметь большую память и высокую производительность, чтобы обеспечивать комфортную работу большого числа пользователей.

Все приложения, работающие в такой архитектуре, полностью находятся в основной памяти хост-ЭВМ. Терминалы являются лишь устройствами ввода-вывода и таким образом в минимальной степени поддерживают интерфейс пользователя.

1. Архитектура "файл-сервер"

Файл-серверные приложения – приложения, схожие по своей структуре с локальными приложениями и использующие сетевой ресурс для хранения программы и данных [5].

* + - Функции сервера: хранения данных и кода программы.
    - Функции клиента: обработка данных происходит исключительно на стороне клиента.

Организация информационных систем на основе использования выделенных файл-серверов все еще является распространенной в связи с наличием большого количества персональных компьютеров разного уровня развитости и сравнительной дешевизны связывания PC в локальные сети.

Конечно, основным достоинством данной архитектуры является простота организации. Проектировщики и разработчики информационной системы находятся в привычных и комфортных условиях IBM PC в среде MS-DOS, Windows или какого-либо облегченного варианта Windows Server. Имеются удобные и развитые средства разработки графического пользовательского интерфейса, простые в использовании средства разработки систем баз данных и/или СУБД.

Достоинства такой архитектуры:

* многопользовательский режим работы с данными;
* удобство централизованного управления доступом;
* низкая стоимость разработки;
* высокая скорость разработки;
* невысокая стоимость обновления и изменения ПО.

Недостатки:

* проблемы многопользовательской работы с данными: последовательный доступ, отсутствие гарантии целостности;
* низкая производительность (зависит от производительности сети, сервера, клиента);
* плохая возможность подключения новых клиентов;

ненадежность системы.

1. Архитектура "клиент-сервер".

Клиент-сервер (Client-server) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением.

Первоначально системы такого уровня базировались на классической двухуровневой клиент-серверной архитектуре (Two-tier architecture). Под клиент-серверным приложением в этом случае понимается информационная система, основанная на использовании серверов баз данных.

На стороне клиента выполняется код приложения, в который обязательно входят компоненты, поддерживающие интерфейс с конечным пользователем, производящие отчеты, выполняющие другие специфичные для приложения функции.

Клиентская часть приложения взаимодействует с клиентской частью программного обеспечения управления базами данных, которая, фактически, является индивидуальным представителем СУБД для приложения.

Заметим, что интерфейс между клиентской частью приложения и клиентской частью сервера баз данных, как правило, основан на использовании языка SQL. Поэтому такие функции, как, например, предварительная обработка форм, предназначенных для запросов к базе данных, или формирование результирующих отчетов выполняются в коде приложения.

Наконец, клиентская часть сервера баз данных, используя средства сетевого доступа, обращается к серверу баз данных, передавая ему текст оператора языка SQL.

Посмотрим теперь, что же происходит на стороне сервера баз данных. В продуктах практически всех компаний сервер получает от клиента текст оператора на языке SQL. Сервер производит компиляцию полученного оператора. Далее (если компиляция завершилась успешно) происходит выполнение оператора.

Разработчики и пользователи информационных систем, основанных на архитектуре "клиент-сервер", часто бывают неудовлетворены постоянно существующими сетевыми накладными расходами, которые следуют из потребности обращаться от клиента к серверу с каждым очередным запросом. На практике распространена ситуация, когда для эффективной работы отдельной клиентской составляющей информационной системы в действительности требуется только небольшая часть общей базы данных. Это приводит к идее поддержки локального кэша общей базы данных на стороне каждого клиента.

Фактически, концепция локального кэширования базы данных является частным случаем концепции реплицированных баз данных. Как и в общем случае, для поддержки локального кэша базы данных программное обеспечение рабочих станций должно содержать компонент управления базами данных – упрощенный вариант сервера баз данных, который, например, может не обеспечивать многопользовательский режим доступа. Отдельной проблемой является обеспечение согласованности (когерентности) кэшей и общей базы данных. Здесь возможны различные решения – от автоматической поддержки согласованности за счет средств базового программного обеспечения управления базами данных до полного перекладывания этой задачи на прикладной уровень.

Преимуществами данной архитектуры являются:

* возможность, в большинстве случаев, распределить функции вычислительной системы между несколькими независимыми компьютерами в сети;
* все данные хранятся на сервере, который, как правило, защищен гораздо лучше большинства клиентов, а также на сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа;
* поддержка многопользовательской работы;
* гарантия целостности данных.

Недостатки:

* неработоспособность сервера может сделать неработоспособной всю вычислительную сеть;
* администрирование данной системы требует квалифицированного профессионала;
* высокая стоимость оборудования;
* бизнес логика приложений осталась в клиентском ПО.

При проектировании информационной системы, основанной на архитектуре "клиент-сервер", большее внимание следует обращать на грамотность общих решений. Технические средства пилотной версии могут быть минимальными (например, в качестве аппаратной основы сервера баз данных может использоваться одна из рабочих станций). После создания пилотной версии нужно провести дополнительную исследовательскую работу, чтобы выяснить узкие места системы. Только после этого необходимо принимать решение о выборе аппаратуры сервера, которая будет использоваться на практике.

В АИС используется двухзвенная клиент-серверная архитектура с технологией толстого клиента. Серверной частью этой архитектуры является сервер базы данных, который выполняет запросы, хранит данные, отслеживает ссылочную целостность, проверку прав и привилегий пользователей. Клиентская часть называется толстой из-за того, что часть бизнес-логики, такая как обработка результатов запросов к базе данных и привидение их к виду, советующему пользовательскому интерфейсу, переносится на сторону клиента, тем самым освобождая сервер БД от части работы.

* 1. Обзор программных средств для реализации системы

Рассмотрим наиболее развитые, популярные CASE-средства.

* Rational Rose
* Silverrun;
* Vantage Team Builder;
* ERwin;
* BPwin;
* S-Designor;
* CASE.Аналитик.
* Rational Rose

1. Rational Rose является самым известным CASE-средством.CASE-средство Rational Rose - продукт Американской фирмы Rational Software Corporation, который автоматизирует этапы проектирования и анализа программного обеспечения. В основе работы данного средства лежит синтез-методология объектно-ориентированного анализа и проектирования.
2. CASE-средство Silverrun разработано американской фирмой CSA (Сomputer Systems Advisers, Inc.). Silverrun ориентирован на спиральную(циклическую) модель жизненного цикла программного продукта. Предназначено для проектирования и анализа ИС бизнес-класса. Структура Silverrun модульная, где каждый модуль является самостоятельным продуктом, что позволяет использовать модули отдельно друг от друга. Данное CASE-средство состоит из 4 модулей:

Достоинство данного CASE-средства в его высокой гибкости, в большом разнообразии изобразительных средств при построении моделей.

Главным недостатком данного CASE-средства является отсутствие жесткого взаимоконтроля компонентов различных моделей между собой.

Silverrun функционирует на платформах MS Windows, Macintosh и OS/2 Presentation Manager. При взаимодействии с другими средствами возможно 3 способа выдачи данных работы во внешние файлы:

Система отчетов. Выдает отчет в форме текстового файла, определив содержимое по репозиторию. Полученный файл можно использовать в дальнейшем: добавить в другой отчет, добавить в текстовый редактор;

Система экспорта/импорта. Имеется возможность не просто форматирования, но и добавления в репозиторий, что позволяет обмениваться информацией с другими CASE средствами, электронными таблицами, текстовыми редакторами, СУБД. Также доступно определение не только содержимое экспортного файла, но и его оформление, т. е. определение маркера начала/конца полей и т. п.;

Хранение репозитория во внешних файлах через ODBC-драйверы. Имеется возможность хранить все проектные данные в формате СУБД.

1. Vantage Team Builder ориентирован на каскадную модель жизненного цикла программного продукта, а также на его поддержку.

У Vantage Team Builder существует несколько различных конфигураций. Конфигурация зависит от используемой СУБД (Informix, Sybase, ORACLE, Ingres), а также от средств разработки приложений (Uniface).

Проектирование ИС с помощью Vantage Team Builder состоит из последовательности 4 стадий (фаз):

* анализ;
* архитектура;
* проектирование;
* реализация.

При таком проектировании результат каждой стадии передается частично или полностью в новую стадию. После переноса результата в новую фазу, связь с предыдущей фазой заканчивается. У Vantage Team Builder имеется возможность многопользовательского доступа к репозиторию проекта. Высокое качество и надежность разрабатываемых продуктов поддерживается благодаря жестоким рамкам в формировании моделей, в структурировании документации.

Имеются реализации Vantage Team Builder основных UNIX-платформ (Solaris, SCO UNIX, AIX, HP-UX) и VMS. Имеется возможность интеграции Vantage Team Builder с другими средствами, т.к. структура репозитория и интерфейса открытые.

1. ERwin, BPwin, S-Designor, CASE.Аналитик

CASE-средства ERwin, BPwin, S-Designor, CASE.Аналитик являются локальными.

ERwin - CASE-средство, которое в качестве методологии использует IDEF1X. Также имеется возможность использовать методологию IE.

IDEF1X была разработана прежде всего для использования в армии США. Широко используется в государственных учреждениях США, в корпорациях финансовых и промышленных направленностей. Методологию IE (Information Engineering) разработали Мартин и Финкельштейн. Она используется большей частью в промышленности.

Выпуск ERwin осуществляется в нескольких конфигурациях, они ориентированны на распространенные средства разработки приложений 4GL.

BPwin - CASE-средство, реализующее в качестве методологии IDEF0. Данный продукт разработан компанией Itd. Logic Works. Довольно развитое средство: позволяет анализировать, документировать и улучшать бизнес-процессы. В данном CASE-средстве поддерживается функциональное моделирование, что дает возможность проводить систематизированный анализ. В BPwin можно создавать как отдельные, так и смешанные модели.

1. S-Designor - CASE-средство фирмы Powersoft, реализующее стандартную методологию. Является средством проектирования реляционных баз данных. По функциональным возможностям близок к CASE-средству ERwin.

В процессе реализации АИС были использованы следующие CASE-средства:

* CA ERwin Process Modeler – инструмент для моделирования, анализа, документирования и оптимизации бизнес-процессов. ERwin Process Modeler можно использовать для графического представления бизнес-процессов. Включает три стандартные методологии: IDEF0 (функциональное моделирование), DFD (моделирование потоков данных) и IDEF3 (моделирование потоков работ).
* Visual Paradigm – набор средств для моделирования информационных систем и бизнес-процессов, генерации кода на базе построенных моделей, проектирования БД и решения многих других задач. Для создания моделей используются такие стандарты и нотации моделирования, как UML.
* MySQL Workbench – инструмент для управления СУБД MySQL, формирования запросов к базам данных через пользовательский интерфейс, создания и последующего экспорта логической модели базы данных в физическую модель с созданием пользователей и назначением им конкретных прав на работу с базой данных. Так же используется для заполнения таблиц базы данных данными.
* React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. ReactJs реализует концепции реактивного программирования: изменение результата работы зависит от состояния компонентов. Так, A = B + C, и результат A всегда будет зависеть от значений B и C. ReactJs постоянно работает с DOM, перерисовывая его при изменении условий (та часть DOM, которую меняет ReactJs, называется компонентом). Ранее подобная практика сильно бы отразилась на производительности приложения, но разработчики ReactJs подошли к решению данного вопроса кардинально: они полностью переписали DOM на Javascript.
  1. Моделирование функционального поведения системы и базы данных

Для моделирования модели процессов работы веб-сервиса виртуальных петиций была использована DFD-диаграмма главного процесса. В приложении Б на рисунке А3.1 представлена функциональная модель использования веб-сервиса.

Модель поведения системы на уровне пользователей представлена в виде Use Case диаграммы в приложении Б на рисунках А3.3. В системе представлено два типа пользователей. Все два типа должны иметь логин и пароль для входа в систему. Пользователь может создавать петицию и просматривать список петиций. Пользователь имеет возможность присоединиться к петиции, завершить её, являясь создателем, и покинуть петицию. Администратор имеет доступ к списку петиций. Администратор так же может проверить петицию на наличие запрещенных материалов, и дальше добавить, либо отклонить её, в список петиций. Все типы пользователей имеют окно личного кабинета, где могут посмотреть информацию о своем аккаунте и сменить никнейм при необходимости.

На рисунке А3.4 в приложении Б приведена структура базы данных, которая представляет из себя логическую модель, которая не привязана к конкретной СУБД. В ней выделяют основные объекты БД и определяют связи между этими объектами. Данная модель построена методом Сущность-связь (Entity Relationship) с использование средства MySQL Workbench.

* 1. Программная реализация системы

Веб-сервисы — это реализация абсолютно четких интерфейсов обмена данными между различными приложениями, которые написаны не только на разных языках, но и распределены на разных узлах сети. Поставленной задачей являлась реализация веб-сервиса для проведения виртуальных петиций на платформе веб-сервисов VK Mini Apps, которая использует фреймворк ReactJS [6], как основу для приложений. Далее будет представлена структура веб-сервиса на таблице 1 и скриншоты практического применения.

Таблица 1 – Структура приложения

|  |  |  |
| --- | --- | --- |
| Файл | Назначение | Функции |
| AddGroupSuccess.js | На рисунке 2 представлено окно успешного добавления в сообщество. | GetGroupInfo  Окно, где любой тип пользователей системы может отследить проведение операции. |
| AddMeetPage.js | На рисунке 3 представлено окно для создания новой петиции в систему. | AddMeet  Самостоятельное создание новой петиции любой группой пользователей. |
| AdminPage.js | На рисунке 4 представлено окно панели администратора. | RemoveComment  В данном окне администратор может модерировать входящие заявки на создание петиций. |
| CommIntegration.js | На рисунке 5 представлено окно с предложением добавить приложение в сообщество. | GetGroupInfo  Добавление приложения в сообщество |
| Favorites.js | На рисунке 6 представлено окно для просмотра петиций, в который вы участвуете. | GetOwneredMeets,  GetUserMeets, GetExpiredUserMeets  В данном окне предполагаются только те петиции, которые пользователь выбрал для участия. |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Home.js | На рисунке 7 представлена домашняя страница веб-сервиса | getStory,GetMeets  Нажав на кнопку «Участвовать», пользователю добавляется петиция в список выбранных. Нажав на кнопку + посередине, пользователь перейдет в окно добавления петиции. Перейдя в правую панель, пользователь откроет окно выбранных петиций. |
| Meet.js | На рисунке 8 представлено окно самой петиции. | GetMeet, GetMeetComments, RemoveMeetMember  Данное окно содержит информацию о создании и комментарии. |
| Offline.js | На рисунке 9 представлено окно оффлайн-режима (Offline.js). | Окно появляется при отсутствии интернет-соединения на устройстве. |
| Onboarding.js | На рисунке 10 представлен стартовый экран веб-сервиса при первом входе (Onboarding.js). | IsFirst  Данное окно содержит информацию о сервисе. |

На рисунке 1 представлена иерархия файлов программы.

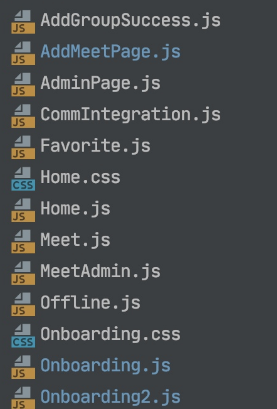


Рисунок 1 – Файлы приложения

Директория программы содержит файлы с классами, которые описывают окна сервиса, файл home.js, в котором содержится домашняя страница веб-сервиса.

Далее представлены реальные скриншоты.

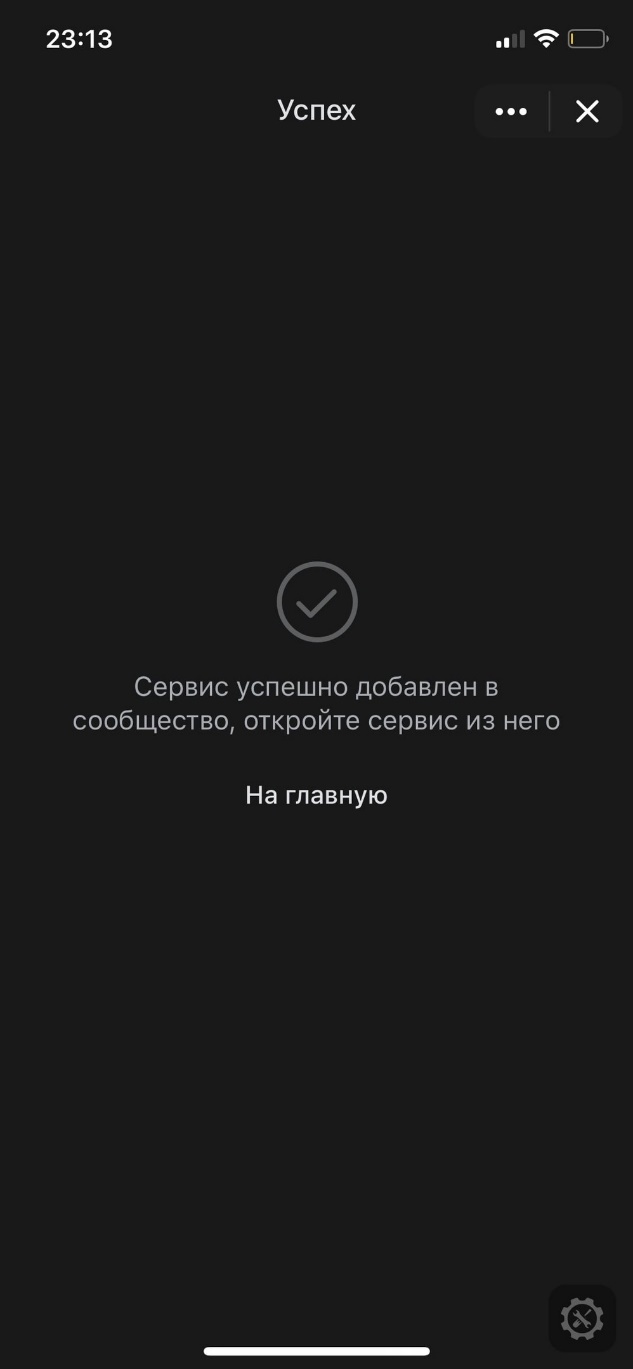


Рисунок 2 – Окно успешного добавления

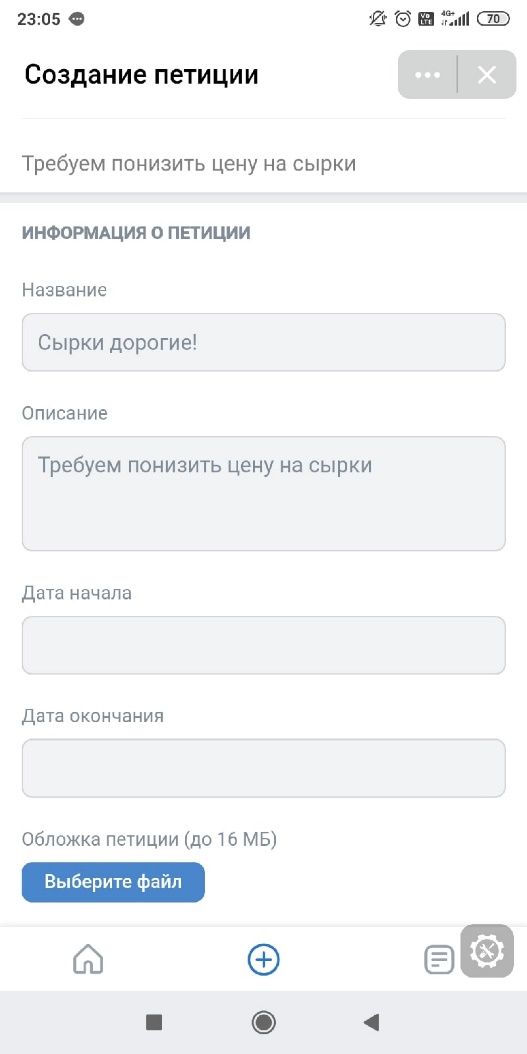


Рисунок 3 – Окно создания



Рисунок 4 – Окно Панель администратора

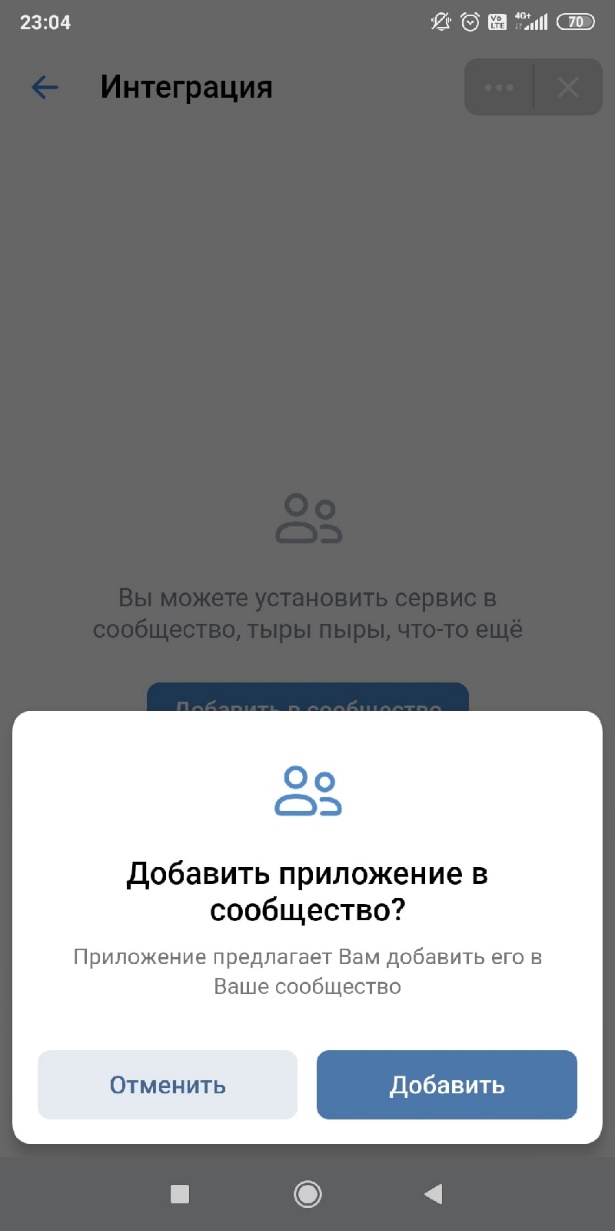


Рисунок 5 – Окно с предложение добавить приложение в сообщество

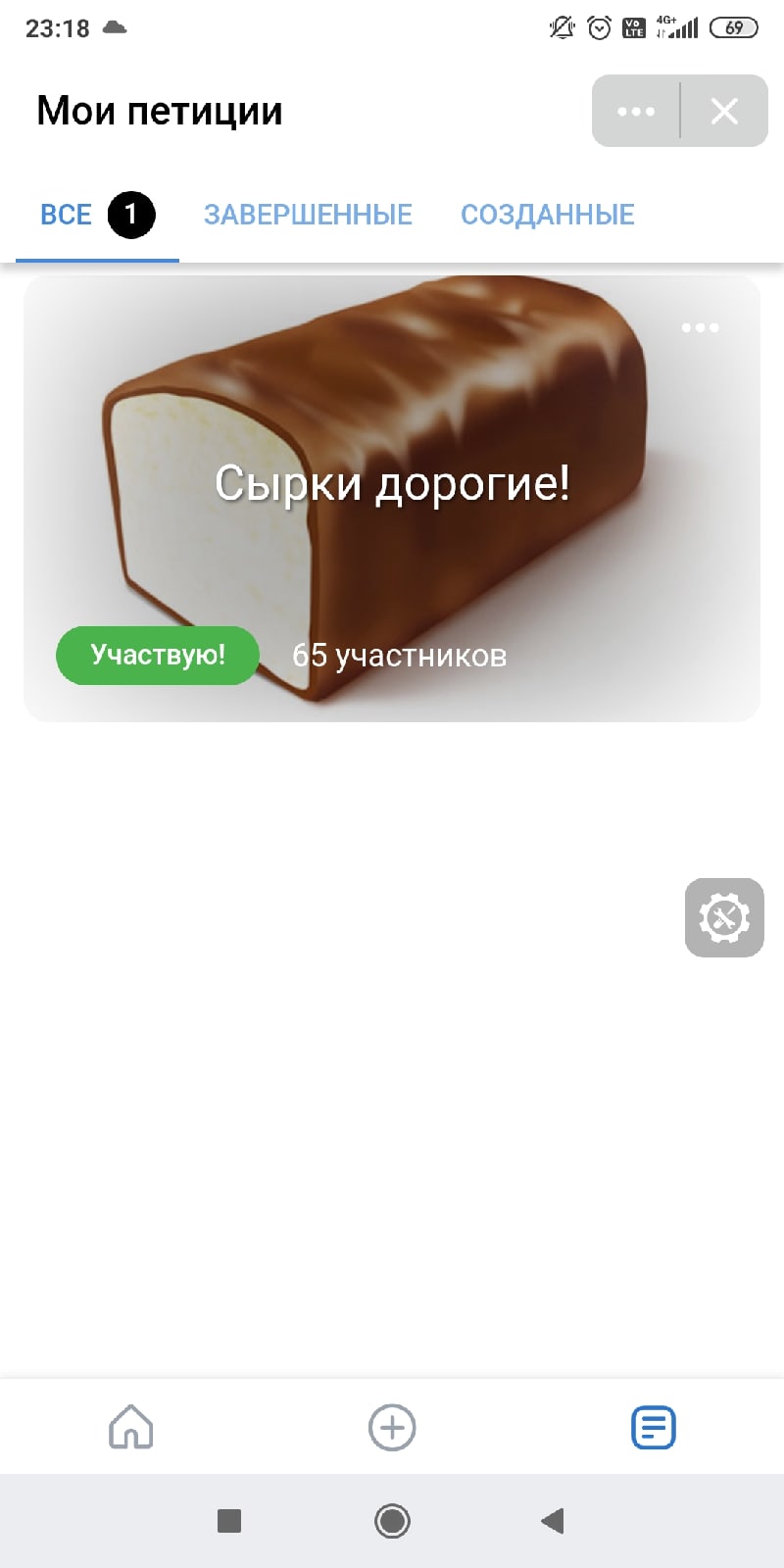


Рисунок 6 – Окно просмотра выбранных петиций

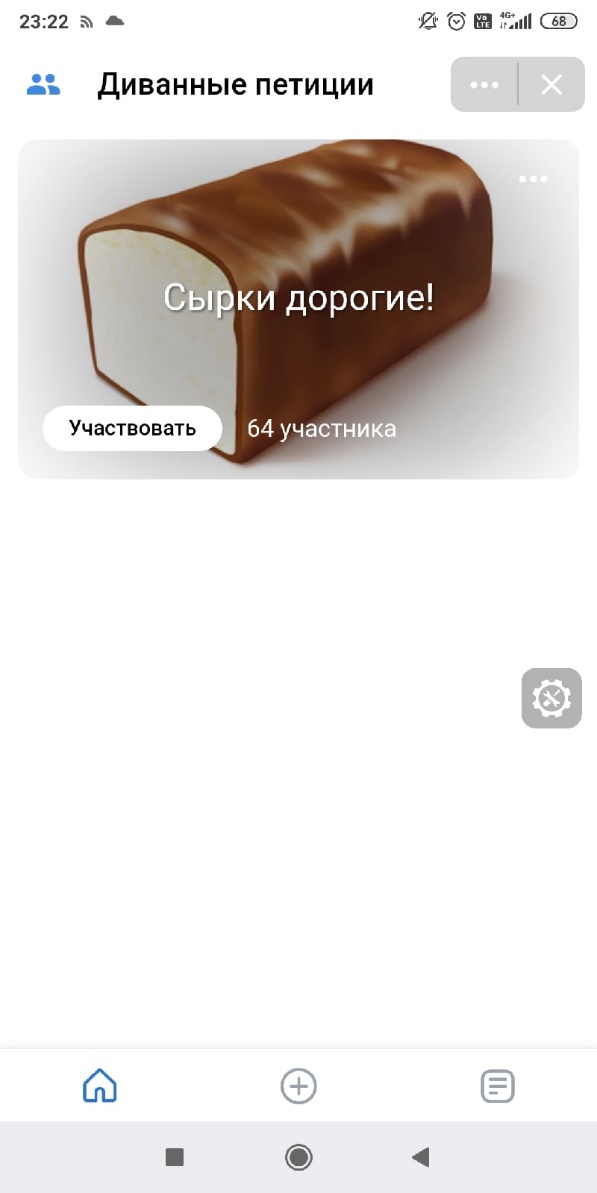


Рисунок 7 – Домашняя страница

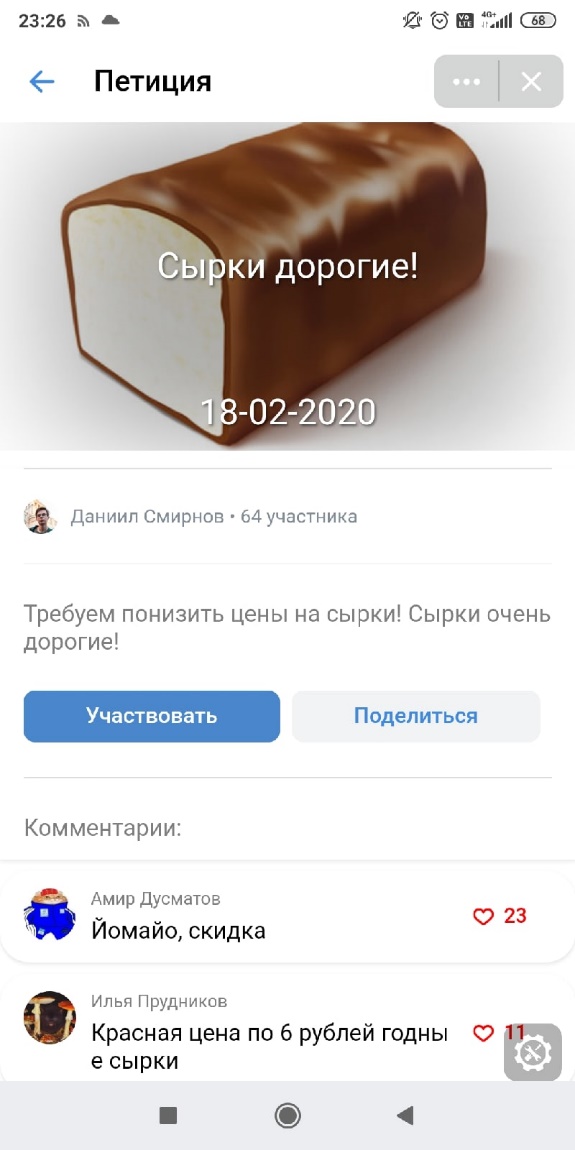


Рисунок 8 – Окно петиции

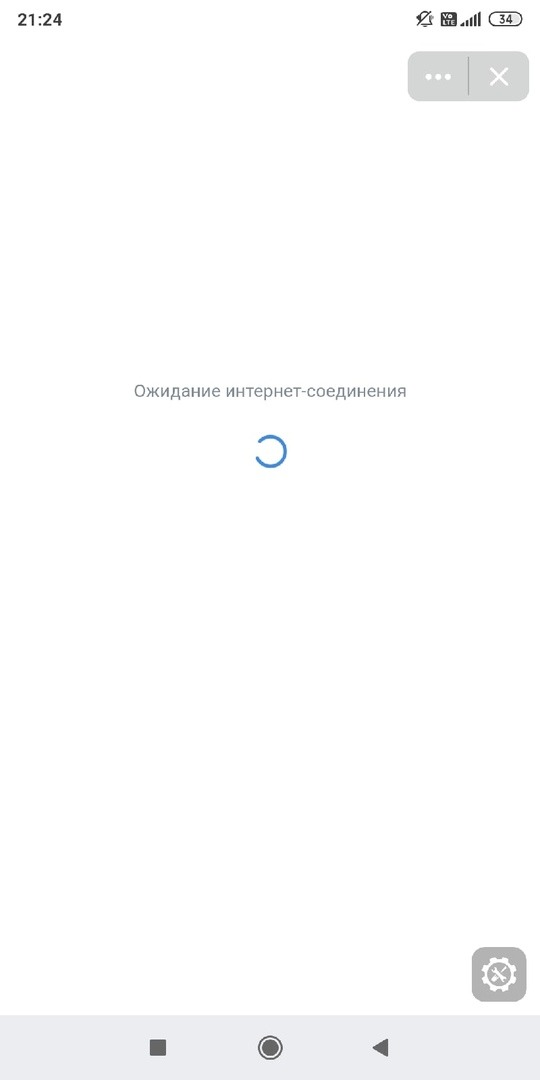


Рисунок 9 – Окно оффлайн-режима



Рисунок 10 – Окно первого входа

ЗАКЛЮЧЕНИЕ

В ходе производственной практики был смоделирована и разработан веб-сервис для проведения виртуальных петиций, имеющий несколько типов пользователей и отдельный интерфейс для каждого из пользователей. В ходе производственной практики были также получены практические навыки моделирования, создания и администрирования баз данных, навыки создания диаграмм в различных нотациях, а также получены навыки разработки пользовательских интерфейсов на фреймворке ReactJS и навыки работы с отчетной документацией.

Все задачи, поставленные перед началом производственной практики, были выполнены.

список источников

1. Иващенко А.В., Сталькин А.А., Калышенко У.М. Применение методологии UML при автоматизации управления бизнес-процессами. Самарский государственный аэрокосмический университет. Научный журнал «Исследовано в России», 2004.
2. Горбаченко И.В., Убиенных Г.Ф., Бобрышева Г.В. Создание функциональной модели информационной системы с помощью CASE-средства CA Erwin Process Modeller 7.3. Учебное пособие. Пенза: ПГУ, 2010. 66 с.
3. Кара-Ушанов В.Ю. Модель «Сущность-связь». Екатеринбург: УрФУ, 2017. [Электронный ресурс] // Портал информационно-образовательных ресурсов Уральского федераьлного университета. URL: <https://study.urfu.ru/Aid/Publication/13604/1/Kara-Ushanov.pdf> (дата обращения: 19.02.2020).
4. История интернета // connect.rin.ru URL: https://connect.rin.ru/articles/internet/2.html (дата обращения: 18.03.20).
5. Hobbes' Internet Timeline // zakon.org URL: https://www.zakon.org/robert/internet/timeline/ (дата обращения: 18.03.20).
6. Обзор документации ReactJS // reactjs.org URL: https://ru.reactjs.org/docs/getting-started.html (дата обращения: 18.03.20).

ПРИЛОЖЕНИЕ А

**Техническое задание**

1. Назначение разработки

Во всем мире, в том числе в Российской Федерации, считается нормой выходить на демонстрационные пикеты и митинги, поддерживая ту или иную идеологию, не противоречащую законам. В настоящее время, согласно статистике с сайта МВД.рф, в нашей стране увеличилось количество оппозиционных митингов. За этим стоит желание высказать в обществе свою позицию и ощутить себя частью движения, которая в силах изменить те или иные аспекты в стране, в которых она имеет сомнения. У каждого человека есть свои мысли по поводу происходящего, но также присутствует и боязнь по поводу отношения властей и людей, имеющих мнение отличное от их. Организация безопасных и законных митингов может упроститься за счет использования веб-сервиса по проведению виртуальных митингов, которые позволит человеку высказать свою позицию в интернете (зайти в чат группы проведения мероприятия, поделиться ссылкой на митинг и прочее), а также поддержать собравшихся, не боясь напряжения ситуации со стороны иных лиц. Таким образом, веб-сервис позволяет провести виртуальный митинг, что приводит к появлению задачи – разработке веб-сервиса на платформе VK Mini Apps.

1. Цель и задачи

Целью проекта разработка веб-сервиса на платформе VK Mini Apps, которая представлена в виде каталога с возможностью запуска веб сервисов в официальных приложениях социальной сети “Вконтакте”. Сервисы разработанные на платформе представляют собой веб-приложения, которое открывается внутри VK Connect browser который, является частью официальных приложений социальной сети “ВКонтакте”, используют API “ВКонтакте”, мобильных устройств, такие как гироскоп, камера, GPS, динамики, вибромотор, вспышка, могут отправлять уведомления и учитывают стандартные сценарии поведения пользователей. Благодаря описанной выше интеграции веб-сервисы на платформе не требуют дополнительной установки, получают данные о профиле пользователя, что позволяет более просто заполнять данные о себе и повысить удобство использования, а также использовать надежный алгоритм авторизации предоставляемый “ВКонтакте”, что обеспечит дополнительную безопасность данных пользователей веб-сервиса.

Назначение разработки – обеспечение граждан средством для организации и проведения митингов без нарушения правопорядка.

Целевой аудиторией приложения являются люди, которых интересует тема пикетирования и участия в митингах и они являются пользователями социальной сети «Вконтакте». Для достижения поставленной цели необходимо выполнить следующие задачи:

Для достижения поставленной цели необходимо выполнить следующие задачи:

* изучить предметную область;
* проанализировать требования к информационному ресурсу;
* спроектировать и разработать базу данных;
* спроектировать и реализовать клиентскую и серверную части ресурса;
* провести отладку и тестирование ресурса.

1. Требования к разработке
2. Требования к функциональным характеристикам

В системе необходимо иметь базу данных для хранения информации и подсистемы ввода-вывода данных, предусматривающих web-интерфейсы для работы пользователя.

В Системе нужно выделить 2 категории пользователей:

* пользователь;
* администратор.

В Системе должны содержаться следующие функциональные модули для упрощенной обработки данных на стороне клиента:

* модуль базы данных, хранящий информацию о пользователях и петициях;
* модуль пользовательского интерфейса, предназначенный для просмотра уже созданных петиций или создания новых;
* модуль интерфейса администратора, предназначенный для рассмотрения созданных петиций пользователем, и последующими возможными действиями (принять, отклонить).

Общий функционал должен обеспечивать следующие возможности в режиме доступа онлайн:

* запуск веб сервисов в официальных приложениях социальной сети “Вконтакте”;
* создание пользователем новых петиций;
* просмотр пользователем все созданные петиции (включая завершенные);
* просмотр пользователем информации о петициях;
* разграничение привилегий и прав для категорий пользователей;
* редактирование созданных петиций для пользователей с привилегиями администратор;
* отклонение на опубликование новых петиций для пользователей с привилегиями администратор;
* публикация новых петиций для пользователей с привилегиями администратор;

1. Разграничения ответственности ролей при доступе

В системе должны быть определены роли пользователей клиент, администратор.

Распределение прав пользователей согласно ролям в системе представлено в таблице 1

Таблица 1 – Распределение прав пользователей согласно ролям

|  |  |  |
| --- | --- | --- |
| Роль  Функции | Клиент | Администратор |
| 1 | 2 | 3 |
| Создание петиций | + | + |
| Просмотр всех созданных петиций | + | + |
| Просмотр информации о петициях | + | + |
| Редактирование созданных петиций | - | + |
| Отклонение на опубликование новых петиций | - | + |
| Публикация новых петиций | - | + |

1. Требования к обработке аварийных ситуаций

Система должна обеспечивать корректную обработку аварийных ситуаций, вызванных неверными действиями пользователей, отсутствие интернета, неверный формат или недопустимые значения входных данных. В указанных случаях пользователю должны выдаваться соответствующие сообщения, после чего возвращаться в рабочее состояние, предшествовавшее неверной команде или некорректному вводу данных.

1. Требования к защите информации от несанкционированного доступа

Система должна поддерживать разграничение прав доступа с возможностью формирования групп пользователей и присвоение группе и каждому пользователю определенных полномочий на доступ к данным системы.

Обеспечение информационной безопасности подсистемы должно удовлетворять следующим требованиям:

* защита от несанкционированного доступа к аккаунтам пользователей;
* обеспечение разделения прав доступа между пользователями.

1. Требования к видам обеспечения
2. Требования к информационному обеспечению

Структура хранения данных системы должна состоять из области постоянного хранения данных, которая должна строиться на основе реляционной модели данных.

1. Требования по применению систем управления базами данных

Для реализации подсистемы хранения данных должна использоваться СУБД MySQL.

1. Требования к лингвистическому обеспечению

При реализации системы должны применяться следующие языки высокого уровня: SQL, JavaScript + фреймворк React с дополнением VKUI.

1. Требования к техническому обеспечению

Пользовательскую часть необходимо реализовать на сервере хостинг–провайдера компании «Вконтакте», который в свою очередь должен удовлетворять следующим критериям:

* дисковое пространство, необходимое для хранения данных системы, – не менее 1 гигабайта;

Серверная часть должна располагаться на сервере, который должен поддерживать бесперебойное подключение к сети.

1. Требования к технологиям и средствам разработки

Разрабатываемое ПО должно быть веб-сервисом. Для разработки информационного обеспечения должна использоваться технология баз данных.

Для разработки должны использоваться следующие программные средства:

* + фреймворк ReactJS;
  + СУБД MySQL.

1. Требования к аппаратному обеспечению

Для работы сервиса на стороне клиента, необходимо иметь установленное приложение ВКонтакте, которое будет соответствовать следующим требованиям:

* Версия ВКонтакте Android должна быть не ниже 5.0.0;
* Версия ВКонтакте iOS должна быть не ниже 4.0.4.

1. Описание входных и выходных данных

Входные и выходные данные Системы должны удовлетворять таблице 2.

Таблица 2

Входные и выходные данные модулей веб-сервиса

|  |  |  |
| --- | --- | --- |
| Функции | Входные данные | Выходные данные |
| 1 | 2 | 3 |
| Создание петиции | Обязательные поля: название, описание, дата начала, дата окончания, согласие с правилами. | Измененная БД |
| Удаление петиции | Обязательные поля: название, описание, дата начала, дата окончания, согласие с правилами. | Измененная БД |
| Отклонение петиции пользователем с привилегией администратора | Идентификатор петиции, описание (название, описание, дата начала, дата окончания, согласие с правилами) | Обновленный список петиций (мои петиции) у пользователя, измененная БД |
| Добавление петиции пользователем с привилегией администратора | Идентификатор петиции, описание (название, описание, дата начала, дата окончания, согласие с правилами) | Обновленный список петиций у пользователя (мои петиции), измененная БД |

1. Модель разработки
2. Функциональная модель

Система должна удовлетворять функциональной модели, представленной в приложении Б, на рисунках А3.1–А3.2.

Роли пользователей в системе должны удовлетворять модели, представленной на диаграммах А3.3. На диаграммах выделены следующие актёры:

* пользователь;
* администратор.

1. Модель данных

База данных должна удовлетворять реляционной модели, представленной в таблицах в приложении Б на рисунке А3.4.

1. Методы тестирования соответствия разработки предъявленным требованиям

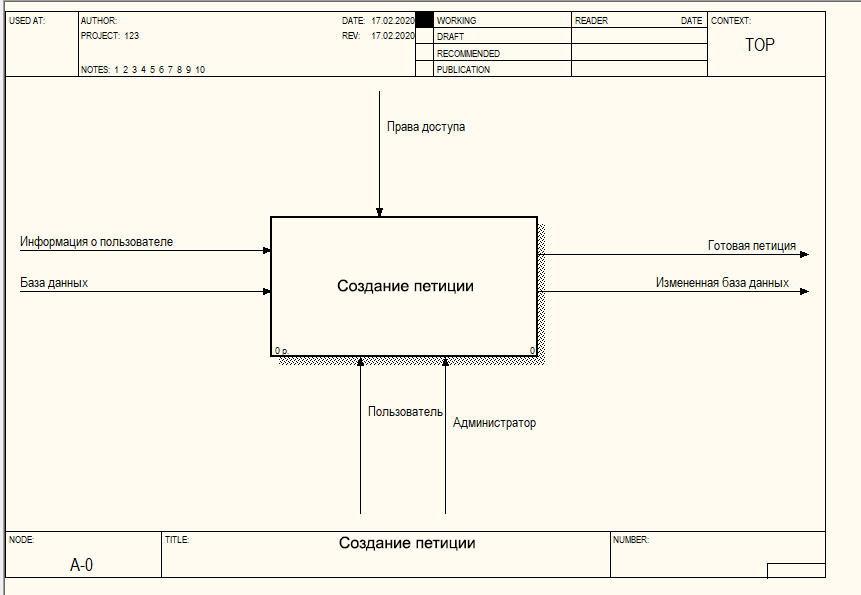
Тестирование должно осуществляться на модульном, интеграционном уровнях.

На этапе разработки программного обеспечения должна применяться технология тестирования «белого ящика» с использованием следующих видов тестирования:

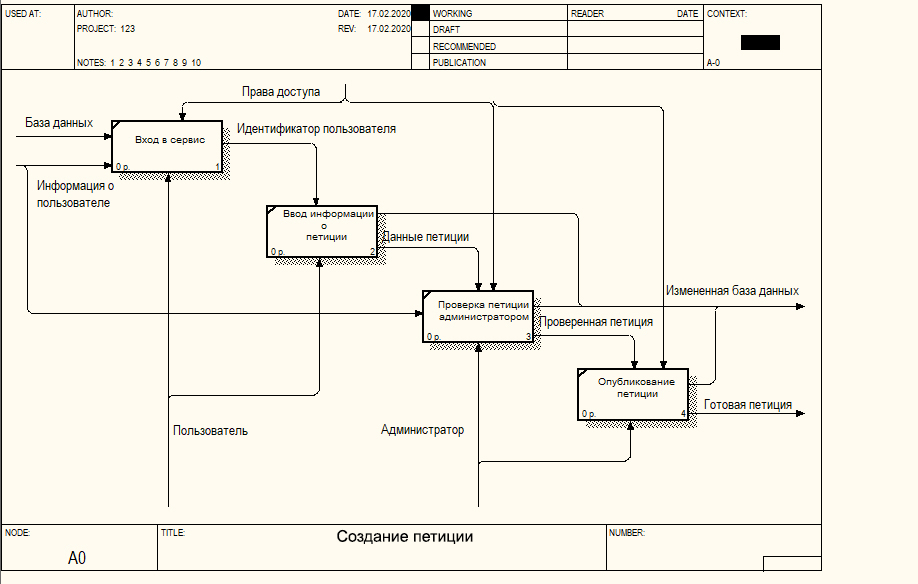
* переходов между состояниями.

ПРИЛОЖЕНИЕ Б

**Модель разработки**



*Рисунок A3.1 – Функциональная модель DFD: главный процесс создания петиции*



*Рисунок A3.2 – Функциональная модель DFD: детализация создания петиции*

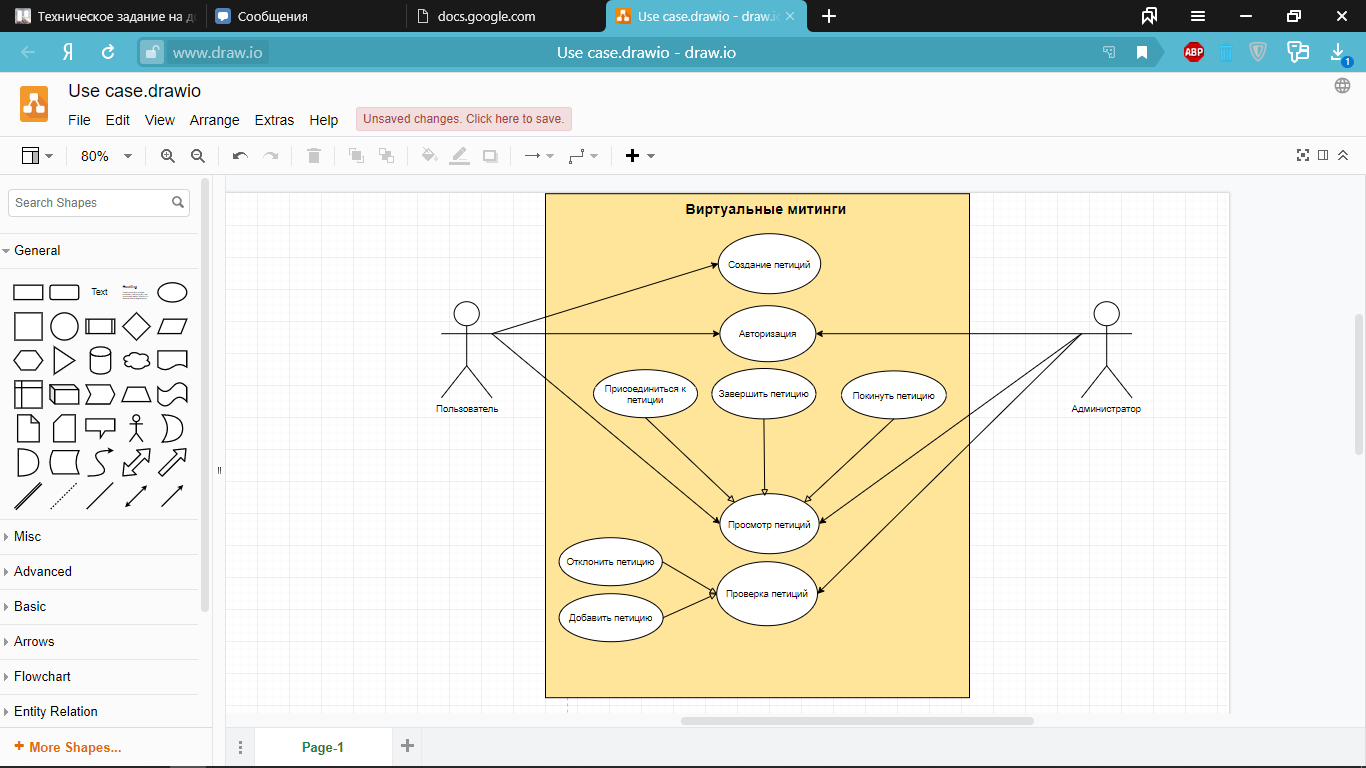


Рисунок A3.3 Поведение пользователей на уровне АИС: администратор, клиент

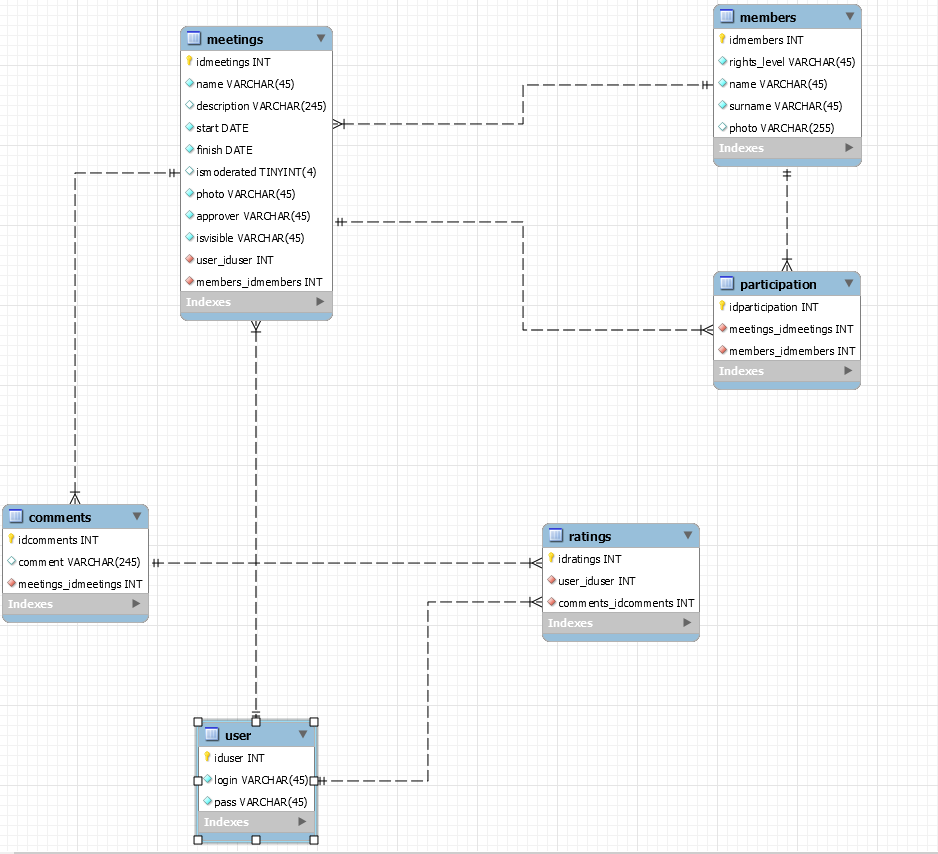


Рисунок A3.4 – Структура базы данных

Полное описание элементов данных, включая ограничения, представлено в таблицах Б1-Б6.

Таблица Б1 – Таблица петиции (meetings)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Обязатель-ность | Первич-ный ключ | Внешний ключ | Ограничения | Пояснения |
| idmeetings | INTEGER | + | + | - | Уникален | Идентификатор  петиции |
| name | VARCHAR (45) | + | - | - | Только буквы | Название петиции |
| description | VARCHAR (245) | + | - | - |  | Подробная информация о петиции |

Продолжение таблицы Б1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| start | DATE | + | - | - | Только дата, без времени | Дата начала петиции |
| finish | DATE | + | - | - | Только дата, без времени | Дата конца петиции |
| ismoderated | TINYINT (4) | + | - | - |  | Флаг модерации петиции |
| photo | VARCHAR (45) | + | - | - | Определенный объем | Обложка петиции |
| approver | VARCHAR (45) | + | - | - | Без знаков | Утверждение петиции |
| isvisible | VARCHAR (45) | + | - | - | Только буквы | Видимость петиции в списке |
| user\_iduser | INTEGER | + | - | + | Уникален | Внешний ключ user |
| members\_idmembers | INTEGER | + | - | + | Уникален | Внешний ключ members |

Таблица Б2 – Таблица участников (members)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Обязатель-ность | Первич-ный ключ | Внешний ключ | Ограничения | Пояснения |
| idmembers | INTEGER | + | + | - | Уникален | Идентификатор  участника |
| rights\_level | VARCHAR (45) | + | - | - | Только буквы | Права пользователя |
| name | VARCHAR (45) | + | - | - | Только буквы | Имя пользователя |
| surname | VARCHAR (45) | + | - | - | Только буквы | Фамилия пользователя |
| photo | VARCHAR (255) | - | - | - | Определенный объем | Фото пользователя |

Таблица Б3 – Таблица участия (participation)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Обязатель-ность | Первич-ный ключ | Внешний ключ | Ограничения | Пояснения |
| idparticipation | INTEGER | + | + | - | Уникален | Идентификатор  участия |
| meetings\_idmeetings | INTEGER | + | - | + | Уникален | Внешний ключ петиции |
| members\_idmembers | INTEGER | + | - | + | Уникален | Внешний пользователя |

Таблица Б4 – Таблица комментариев (comments)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Обязатель-ность | Первич-ный ключ | Внешний ключ | Ограничения | Пояснения |
| idcommetns | INTEGER | + | + | - | Уникален | Идентификатор  комментария |
| comment | VARCHAR (245) | + | - | - |  | Комментарий |
| meetings\_idmeetings | INTEGER | + | - | + | Уникален | Внешний ключ петиции |

Таблица Б5 – Таблица пользователей (user)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Обязатель-ность | Первич-ный ключ | Внешний ключ | Ограничения | Пояснения |
| iduser | INTEGER | + | + | - | Уникален | Идентификатор  пользователя |
| login | VARCHAR (45) | + | - | - | Без символов, уникален | Логин пользователя |
| pass | VARCHAR (45) | + | - | - |  | Пароль пользователя |

Таблица Б6 – Таблица рейтинга комментариев (ratings)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Обязатель-ность | Первич-ный ключ | Внешний ключ | Ограничения | Пояснения |
| idratings | INTEGER | + | + | - | Уникален | Идентификатор  пользователя |
| user\_iduser | INTEGER | + | - | + | Уникален | Внешний ключ пользователя |
| comments\_idcomments | INTEGER | + | - | + | Уникален | Внешний ключ комментария |