МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
« » 2018 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка базы данных школы и приложения для платформы Android

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнил Ким А.Г. Группа ВТ-14-2
Научный руководитель к. ф.-м. н. доцент Шайхин Б. М.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«22» мая 2018 г.

по безопасности жизнедеятельности: ст. преп. _____ А.А. Абикенова
«23» 05 2018 г.

по применению
вычислительной техники: ст. преп. _____ А.М. Рамазанова
«22» 05 2018 г.

Нормоконтролер: PhD, ст. преп. _____ Ж. Бидахмет
«31» 05 2018 г.

Рецензент: к.т.н., доцент _____ Ескендирова Д.М.
«____» _____ 2018 г.

Алматы 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Ким Алексею Геннадьевичу

Тема работы: Разработка базы данных школы и приложения для платформы Android

Утверждена приказом по университету № 155 от «23» октября 2017 г.

Срок сдачи законченной работы «1» июня 2018 г.

Исходные данные к работе (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): MySQL – среда разработки БД, Java - среда программирования, Android Studio – интегрированная среда разработки.

Перечень вопросов, подлежащих разработке в дипломной работе, или краткое содержание дипломной работы:

а) разработка базы данных для школы;
б) методы разработки базы данных;
в) технологии разработки мобильных приложений;
г) вопросы безопасности жизнедеятельности и охраны труда;
д) экономическая эффективность работ по стандартизации.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 14 таблиц, 42 иллюстрации.

Основная рекомендуемая литература:

1 Орлов Г.Г. Инженерные решения по охране труда в строительстве. Справочник строителя (1985), коричневое издение, под редакцией г. г.Оорлова, Москва, строиздат 1985

2 Распределенная обработка данных: курс лекций / Сост. Найханова Л.В. – Улан-Удэ, Издательство ВСГТУ, 2001. – 122 с.

3 Шилдт Г. Java, Полное руководство. Издательство Вильямс, 8-е издание, 2012.

4 Шилдт Г. Java, Руководство для начинающих. 5-е издание. Издательство Вильямс, 2012. – 624 с.

Консультации по работе с указанием относящихся к ним разделов работы

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	18.03.2018-22.05.2018	
Безопасности жизнедеятельности	Абикенова А.А.	14.03.2018-23.05.2018	
Программная часть	Рамазанова А.М.	05.05.2018-22.05.2018	
Нормоконтролер	Бидахмет Ж.	25.05.2018-31.05.2018	

График
подготовки дипломной работы

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Анализ предметной области	15.01.2018-15.02.2018	
Разработка и реализация структуры базы данных	15.02.2018-25.03.2018	
Создание приложения для платформы Android	25.03.2018-01.05.2018	

Дата выдачи задания «25» октября 2017 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель работы _____ Б.М. Шайхин

Задание принял к исполнению студент _____ А.Г. Ким

Аңдатпа

Бұл дипломдық жобада №105 Халықаралық лингвистикалық гимназиясы үшін деректер базасын жобалау технологиясы қарастырылған. MySQL Database және Java технологияларын қолдануымен деректер базасы мен пайдаланушы интерфейсін құру процесі көрсетілген. ИСР платформа әзірлеушісі Google Android болғандықтан, әзірлеу ортасы Android Studio болып таңдалды.

Осы қосымшаның функционалдығы ата-аналар туралы ақпаратты жинауды жеңілдетеді, оңайлатады, уақытылы өзгерістер жасайды және ақпараттың өзектілігін бақылайды. Мобильді қосымшаға ата-аналарға арналған қоңыраулардың кестесін қарау мүмкіндігі, және оқушылар мен ата-аналарға арналған кестені қарау функциясы қосылған.

Өмір қауіпсіздігі тарауында эвакуация маршруттары мен қозғалысы есептеледі және төтенше жағдайдағы эвакуация құрылымы құрастырылған.

Аннотация

В данном дипломном проекте рассматриваются технологии разработки базы данных для КГУ Гимназии №105. Показан процесс создания БД и пользовательского интерфейса, с использованием MySQL Database и языка программирования Java. Средой разработки была выбрана Android Studio, так как создателем данной ИСР является сам разработчик платформы Google Android.

Функционал данного приложения заметно сможет облегчить и упростить сбор информации о родителях, своевременно вносить изменения и следить за актуальностью информации. В приложение встроена возможность просмотра расписания звонков для родителей, а также функция просмотра расписания для учеников и родителей.

В разделе безопасность жизнедеятельности произведен расчет эвакуационных путей и составлена полная схема эвакуации при чрезвычайных ситуациях.

Annotation

This Diploma Project involves considering the database development technologies for Municipal Public Institution gymnasium № 105. The process of the database and human interface creation using the MySQL Database and the Java programming language. Android Studio was chosen as integrated development environment, because creator of this IDE is the developer of the Google Android platform.

The functionality (features) of this application (application software) can significantly facilitate and simplify the process of personal information (data) collection, make timely (up to date) changes and monitor the relevance of information. There is built – in feature, that gives ability to check timetable, purposed for parents, and the mobile application includes schedule view for pupils.

In the health and safety (vital activity security) section, the emergency exit route was calculated, and evacuation scheme is drawn up in emergency situations.

Содержание

Введение 7
1 Теоретическая часть 11
1.1 Понятие БД и СУБД 11
1.2 Классификация баз данных 12
1.3 Реляционная база данных 14
1.4 Нормализация отношений 14
1.5 Типы отношений 15
1.6 Платформа WAMP 19
2 Разработка и реализация базы данных 22
2.1 Разработка технического задания 22
2.2 Подготовка к разработке и предустановка программного обеспечения 22
2.3 Разработка структуры базы данных 24
2.4 Составление реляционных отношений 26
2.5 Средства разработки базы данных 28
2.6 Разработка связи с базой данных 32
3 Разработка пользовательского графического интерфейса приложения 38
3.1 Установка Android Studio, первичная настройка и установка необходимых компонентов для полного функционирования среды. 38
3.2 Создание и сборка проекта в Android Studio 48
3.3 Тестирование программы 53
4 Технико-экономическое обоснование 60
4.1 Расчет трудоемкости разработки ПО. 60
4.2 Расчет затрат на разработку информационных технологий 64
4.3 Расчет цены программного продукта 68
4.4 Вывод по технико – экономической части 69
5 Безопасность жизнедеятельности 71
5.1 Анализ условий труда в здании КГУ гимназии №105 71
5.2 Расчет эвакуационных путей. 73
5.3 Полная схема движения при эвакуации 78
5.4 Вывод по разделу безопасность жизнедеятельности 79
Заключение 80

Список литературы 81
Приложение А **Ошибка! Закладка не определена.**

Введение

Образование – фундаментальная составляющая любого развитого общества. Именно образование является краеугольным камнем в жизни каждого человека. Школа – первый жизненный институт, который прививает нам любовь к знаниям и помогает определиться с тем, кем мы хотим стать в будущем.

Работа в школе подразумевает не только обучающий процесс, но и скрывает в себе множество дополнительных обязанностей, таких как сбор различных взносов, сбор различной информации, ведение журналов и дневников, составление поурочных планов, организация внеурочных мероприятий, слежение за дисциплиной в роли классного руководителя и не другое.

В ходе исследования учебного заведения – коммунальное государственное учреждение гимназия №105, было выяснено, что преподавательский состав использует относительно молодую систему учета и контроля за успеваемостью – «Kundelik.kz». Однако вся документация о родителях и учениках хранится в бумажном виде, либо в документах на жестких дисках, что не является удобным, безотказным и быстродоступным вариантом.

Цель данной дипломной работы – разработка и проектирование базы данных и приложения, для просмотра, добавления и поддержания актуального статуса информации, а также облегчить процесс сбора документов при поступлении и ежегодный процесс обновления и актуализации информации.

Для решения вышеуказанной цели были поставлены следующие задачи: исследовать предметную область в сфере среднеобразовательных учреждений, изучить текущий процесс сбора информации, создать базу данных «Коммунального государственного учреждения гимназии № 105», разработать мобильное приложения для удобства использования базы данных.

Платформа для разработки мобильного приложения – Android, так как в ходе опроса было выявлено, что более 80% родителей являются обладателями смартфонов с операционной системой на базе Android.

1 Теоретическая часть

1.1 Понятие БД и СУБД

База данных представляет собой специальный набор организованных данных, хранящихся в памяти компьютерной системы, показывающий состояние вещей и их взаимосвязи между собой.

Следует отметить, что это определение не является единственным возможным значением. Информация об определениях часто не совсем соответствует математическому совершенству. Если вы посмотрите на представление «база данных» с точки зрения пользователя, будут другие значения: база данных представляет собой набор данных, хранящихся в данных конкретной компании.

Например, в базе данных предприятия вы можете сохранить следующее:

– вся информация о рабочей силе, сотрудниках и сотрудниках компании;
– информация о материальных ценностях;
– данные о поступлении сырья и компонентов;
– информация о запасах;
– данные о выпуске готовой продукции;
– заказы и заказы от директоров и т. Д.

Даже незначительные изменения во всей информации могут привести к серьезным изменениям в других областях.

Пример. Объявление о найме сотрудника приводит к тому, что он не только изменяет личные данные сотрудника, но также изменяет их списки рабочих единиц, задачи расчета заработной платы, отпуска и т. Д. D.

Поскольку база данных основана на информационной структуре, база данных делится на три типа: плоскую (реляционную), сетевую и иерархическую.

Опыт работы с базами данных позволяет определить общий набор функций производительности:

– полнота – чем больше информации в базе, тем больше шанс, что в ней содержится необходимая информация (но, лучше избегать захламленности и перегруженности информацией базы);

– верно составленная структура – чем лучше составлена и продумана база данных, тем проще и быстрее найти в ней требуемые данные;

– актуальность – для того чтобы база данных была полной и точной в любой момент запроса информации, необходима возможность постоянного и оперативного обновления;

– комфорт использования – одним из важных аспектов успешной и удобной базы является простота и удобство в использовании, для этого база должна иметь развитые методы доступа к любому блоку информации;

Система управления базами данных (СУБД) - это программное обеспечение для создания и редактирования баз данных, для отображения и

получения информации о них. В технологиях обработки базы данных делятся на централизованные и распределенные. Централизованная база данных хранится в памяти вычислительного устройства.

Распределенная база данных состоит из множества компонентов, хранящихся в одной компьютерной сети, но на некоторых компьютерах. Работа в базе данных этого типа выполняется с использованием системы управления распределенной базой данных, СУРБД.

Централизованные методы доступа к базе данных подразделяются на:

– базы данных с локальным доступом (данные и обработка, хранящиеся на компьютере);

– с удаленным доступом (сеть). RDBMS можно установить удаленно, используя архитектуру файлового сервера и клиентский сервер.

Файл серверной архитектуры. Принцип организации: машина назначается центральным офисом (файловым сервером), где хранится централизованная база данных. Остальные сетевые машины выполняют функции рабочей станции. Файлы базы данных переносятся на эти диски и обрабатываются там в соответствии с запросом пользователя с рабочей станции. Эффективность таких систем не может оставаться на том же уровне и уменьшается, когда требуется интенсивный параллельный доступ к тем же данным.

Клиент-серверная архитектура. Организационные принципы. Центральная машина (сервер базы данных) хранит централизованную базу данных и программу обработки. Клиент отправляет запрос, который обрабатывается сервером и отправляется клиенту на основе данных, полученных по запросу.

1.2 Классификация баз данных

Классифицирование базы данных по типу хранимых данных

База данных документов, сгруппированных в разные группы атрибутов (организации), классифицируется как база данных документов.

Документ представляет собой текстовый документ или ссылку. Документальная база данных разделена на текстовые файлы, тезисы (рефераты) и библиографии. Это разделение не так важно, как способ хранения информации. Ниже приведен следующий раздел: Хранилище базы данных исходных документов или ссылочных типов, которое вы можете увидеть в исходном документе.

– фактографические БД объединяют данные по факту совершения события (дата выпуска товара, год рождения сотрудника).

– лексикографические БД объединяют словари, классификаторы, и т.д. документы.

Типичным примером является то, что база данных документов может использоваться как база данных для контроля «формальных» составных документов. Вы наверняка сталкивались с таким документом. Например, в

паспортном отделении или отделе кадров вы заполняли «форму №» или «документ» в этой форме.

Классификация баз данных по обращению к ним

Лично используемые базы данных классифицируются как личные или локальные базы данных.

– интегрированные, иначе централизованные базы данных позволяют осуществлять коллективный доступ к данным. Этот доступ может быть многопользовательским (все одновременно) и параллельным (независимым);

– распределительные базы данных аналогичны интеграции баз данных, но они могут быть физически разделены на разные машины и логически обрабатывают их в целом.

Вышеупомянутые категории не особенно интересны для пользователя. Для пользователей классификация методов анализа данных и типы представляющих интерес моделей, которые изображены на рисунке 1.1.

.

Рисунок 1.1 – Классификация баз данных

1.3 Реляционная база данных

Реляционная база данных представляет собой набор взаимосвязанных таблиц, каждый из которых содержит информацию об определенном типе объекта. Строка таблицы содержит данные об объекте (например, продукт, клиент) и в столбце массивов различных свойств этих объектов - атрибуты (например, имя, код продукта, информация о клиенте). В записях, т. Е. Строки таблиц имеют одинаковую структуру. Каждое поле (т.е. Столбец) описывает только одну особенность объекта и имеет строго определенный характер. Все записи имеют одинаковые поля, только они отображают разные информационные атрибуты объекта.

В реляционной базе данных каждый из них находится в поле, которое однозначно идентифицирует каждую строку в таблице. Если ключ состоит из нескольких полей, он называется составным полем. Ключ должен быть уникальным и уникальным образом идентифицировать запись. С ключевым словом вы можете найти запись. Ключ также используется для организации информации в базе данных.

Таблица реляционной БД должна быть нормализована. Нормализация отношений – формальные ограничения на таблицы, избежание дублирования и обеспечение согласованности данных, хранящихся в базе данных, уменьшают объем работы, необходимой для поддержания базы данных реляционной нормализации для данных. Таблицы реляционных баз данных. Нормализация отношений - это формальное ограничение в таблице, устранение дублирования, обеспечение согласованности данных, хранящихся в базе данных, и сокращение объема работы, необходимой для поддержания базы данных.

Пусть создана таблица Студент, которое содержит следующие поля: номер группы, имя, номер зачетке, дата рождения, профессиональное имя, название факультета. У некоторых пользователей есть недостатки:

– дублирование информации (наименование специальности и факультета повторяются для каждого студента), следовательно, увеличится объем БД;

– процедура обновления информации в таблице затрудняется из-за необходимости редактирования каждой записи таблицы.

1.4 Нормализация отношений

Нормализация таблиц предназначена для устранения этих недостатков. Имеется три нормальные формы отношений.

Первая нормальная форма. Реляционная таблица приведена к первой нормальной форме тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто. Так, если из таблицы Студент требуется получать сведения по имени студента, то поле ФИО следует разбить на части Фамилия, Имя, Отчество.

Вторая нормальная форма. Реляционная таблица задана во второй нормальной форме, если она удовлетворяет требованиям первой нормальной формы и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом. Чтобы привести таблицу ко второй нормальной форме, необходимо определить функциональную зависимость полей. Функциональная зависимость полей — это зависимость, при которой в экземпляре информационного объекта определенному значению ключевого реквизита соответствует только одно значение описательного реквизита.

Третья нормальная форма. Таблица находится в третьей нормальной форме, если она удовлетворяет требованиям второй нормальной формы, ни одно из ее не ключевых полей не зависит функционально от любого другого не ключевого поля.

Над реляционными таблицами возможны следующие операции:

– объединение таблиц с одинаковой структурой. Результат – общая таблица: сначала первая, затем вторая (конкатенация);

– пересечение таблиц с одинаковой структурой. Результат – выбираются те записи, которые находятся в обеих таблицах;

– вычитание таблиц с одинаковой структурой. Результат – выбираются те записи, которых нет в вычитаемом;

– выборка (горизонтальное подмножество). Результат – выбираются записи, отвечающие определенным условиям;

– проекция (вертикальное подмножество). Результат – отношение, содержащее часть полей из исходных таблиц;

– декартово произведение двух таблиц. Записи результирующей таблицы получаются путем объединения каждой записи первой таблицы с каждой записью другой таблицы;

Реляционные таблицы могут быть связаны друг с другом, следовательно, данные могут извлекаться одновременно из нескольких таблиц. Таблицы связываются между собой для того, чтобы в конечном счете уменьшить объем БД. Связь каждой пары таблиц обеспечивается при наличии в них одинаковых столбцов.

1.5 Типы отношений

Существуют следующие типы информационных связей:

– один-к-одному;

– один-ко-многим;

– многие-ко-многим.

Связь один-к-одному предполагает, что одному атрибуту первой таблицы соответствует только один атрибут второй таблицы и наоборот.

Связь один-ко-многим предполагает, что одному атрибуту первой таблицы соответствует несколько атрибутов второй таблицы.

Связь многие-ко-многим предполагает, что одному атрибуту первой таблицы соответствует несколько атрибутов второй таблицы и наоборот.

База данных состоит из различных объектов, таких как таблицы, виды, домены, сохраненные процедуры, триггеры. Объекты базы данных содержат всю информацию о ее структуре и данных. Объекты базы данных так же упоминаются, как метаданные.

Следующие разделы содержат краткую информацию об объектах и концепциях базы данных InterBase:

- таблицы (Tables);
- столбцы (Columns);
- типы данных (Data types);
- домены (Domains);

Справочные ограничения целостности (Referential integrity constraints)

- индексы (Indexes);
- виды (Views);
- сохраненные процедуры (Stored procedures);
- триггеры (Triggers);
- генераторы (Generators);
- защита (Security);
- таблицы (Tables);

Реляционные базы данных хранят все данные в таблицах. Таблица — это структура, состоящая из множества неупорядоченных горизонтальных строк (rows), каждая из которых содержит одинаковое количество вертикальных столбцов (colums). Пересечение отдельной строки и столбца называеися полем (field), которое содержит специфическую информацию. Многие принципы работы реляционной базы данных взяты из определений отношений (relations) между таблицами.

InterBase хранит информацию о метаданных в специальных таблицах, которые называются системными таблицами (system tables). Системные таблицы имеют специальные столбцы, которые содержат информацию о типе метаданных в этой таблице. Имена всех системных таблиц начинаются с "RDB$". Пример системной таблицы - RDB$RELATIONS, которая содержит информацию о каждой таблице в базе данных.

Системные таблицы имеют такую же структуру, как и определенные пользователем таблицы и расположены в той же самой базе. Так как метаданные, пользовательские таблицы, и данные все вместе расположены в одном и том же файле базы данных, каждая база данных является законченным модулем и может быть легко перенесена между различными машинами.

Системные таблицы могут быть изменены подобно любой другой таблице базы данных. Если вы не понимаете всех взаимосвязей между системными таблицами, то непосредственное изменение их может иметь негативный эффект на другие системные таблицы и разрушить вашу базу данных.

Создание таблицы главным образом подразумевает определение столбцов таблицы. Главные атрибуты столбца включают:

– имя столбца;

– тип данных столбца или домен, на котором он базируется;

– может или нет поле столбца принимать значение NULL;

– факультативно справочные ограничения целостности (referential integrity constraints).

Данные сохранены в определенном формате, который называется типом данных (data type). Типы данных могут быть классифицированы по четырем категориям: числовые (numeric), символьные (character), даты (date) и BLOB. Числовые данные включают в себя все числа, начиная с целых вплоть до чисел двойной точности с плавающей точкой. Символьные данные содержат строки текста. Даты используются для хранения дат и времени.

В то время как числовые, символьные и даты являются стандартными типами данных, BLOB-тип заслуживает специального внимания.

InterBase поддерживает такой тип данных, как большие бинарные объекты (binary large object - BLOB), которые могут хранить данные неограниченного размера. Тип BLOB это расширение стандартной реляционной модели, которая обычно обеспечивает только типы данных фиксированной длины.

Тип данных BLOB аналогичен последовательному файлу (flat file), BLOB данные могут быть сохранены в любом формате (к примеру, бинарном или ASCII). BLOB, однако, это не отдельный файл. BLOB данные хранятся в базе данных наряду со всеми другими данными. Так как BLOB столбцы часто содержат большие и переменные объемы данных, BLOB столбцы хранятся в отдельных сегментах.

InterBase не поддерживает непосредственно преобразование BLOB данных в другие форматы, но на некоторых платформах, BLOB фильтры могут транслировать BLOB данные из одного формата в другой.

В добавление к явному определению типа данных столбцов, InterBase обеспечивает глобальные определения столбцов или домены (domains), на которых могут базироваться определения столбцов. Домен содержит информацию о тип данных, устанавливает атрибуты и ограничения целостностистолбцов. В последующем при создании таблиц возможно использовать домены для определения столбцов.

Справочные ограничения целостности (Referential integrity constraints)

InterBase позволяет вам определять правила, обеспечивающие целостность информации, хранящейся в столбцах, эти прравила названы справочными ограничениями целостности (referential integrity constraints). Ограничения целостности управляют связями типа столбец-таблица (column-to-table) и таблица-таблица (table-to-table) а также проверкой ввода данных. Они выпонены через первичные ключи (primary keys), внешние ключи (foreign keys) и проверочные ограничения (check constraints). Обычно первичный ключ — это столбец (или группа столбцов), которые используются, чтобы уникально

идентифицировать строку таблицы. Внешний ключ — это столбец, чьи значения должны соответствовать значениям столбца в другой таблице. Проверочные ограничения - ограничивают ввод данных определенным диапазоном или набором значений.

Индексы – это механизм для улучшения быстродействия поиска данных. Индекс определяет столбцы, которые могут быть использованы для эффективного поиска и сортировки в таблице.

InterBase автоматически определят уникальные индексы для первичных и внешних ключей таблицы.

Вид (view) это виртуальная таблица, которая не сохранена физически в базе данных, но ведет себя точно также как "реальная" таблица. Вид может содержать данные из одной или более таблиц, или других видов и используется для хранения часто используемых запросов (queries) или множества запросов в базе данных.

Виды могут также обеспечивать ограниченные средства защиты, так как они могут обеспечивать доступ пользователей к подмножеству доступных данных при скрытии других связанных и чувствительных данных.

Сохраненные процедуры (stored procedure) это отдельные программы, написанные на языке процедур и триггеров InterBase, который является расширением SQL. Сохраненные процедуры являются частью метаданных базы данных. Сохраненные процедуры могут получать входные параметры, возвращать значения приложению и могут быть вызваны явно из приложения или подстановкой вместо имени таблицы в инструкции SELECT.

Сохраненные процедуры обеспечивают следующие возможности:

– модульный проект: сохраненные процедуры могут быть общими для приложений, которые обращаются к той же самой базе данных, что позволяет избегать повторяющегося кода, и уменьшает размер приложений;

– упрощают сопровождение приложений: при обновлении процедур, изменения автоматически отражаются во всех приложениях, которые используют их без необходимости их повторной компиляции и сборки;

– улучшают эффективность работы: Особенно для удаленных клиентов. Сохраненные процедуры выполняются сервером, а не клиентом, что снижает сетевой трафик;

Триггеры – это отдельная программа, ассоциированная с таблицей или видом, которая автоматически выполняет действия, при добавлениях, изменений или удалений строки в таблице или виде.

Триггеры могут обеспечивать следующие возможности:

– автоматическое ограничение ввода данных, чтобы гарантировать, что пользователь ввел только допустимые значения в поля столбцов;

– упрощение сопровождения приложений, так как изменение в триггере автоматически отражается во всех приложения, которые используют таблицы со связанными с ними триггерами;

– автоматическое документирование изменений таблицы. Приложение может управлять логом изменений с помощью триггеров, которые выполняются всякий раз, когда происходит изменение таблицы.

Когда триггер вызван, он имеет непосредственный доступ к добавлению, изменению или уничтожению данных. Триггеру могут быть так же доступны данные из других таблиц. Вы можете разрабатывать триггеры для:

– завершения операции, возможно с сообщением об ошибке;
– установки значений в записи, которой вы обращаетесь;
– добавления, изменения или удаления строк в других таблицах.

Генератор (generator) это механизм который создает последовательный уникальный номер, который автоматически вставляется в столбец базой данных, когда выполняются операции INSERT или UPDATE. Генератор обычно применяется для создания уникальных значений, вставляемых в столбец, который используется как PRIMARY KEY. Для базы данных может быть определено любое число генераторов, каждый генератор должен имеет уникальное имя.

SQL защита (securite) управляется на уровне таблицы привилегий доступа - списка операций, которые разрешены пользователю над данной таблицей или видом. Инструкция GRANT назначает привилегии доступа к таблице или виду конкретным пользователям или процедурам. Инструкция REVOKE удаляет предварительно предоставленные привилегии доступа.

1.6 Платформа WAMP

WAMP это платформа, объединяющая программные продукты для веб-строительства локального сайта и разработки веб приложений в среде Windows. WAMP объединяет ОС Windows с готовой связкой веб-сервера Apache + система управления базами данных (СУБД)+ интерпретатор PHP для работы с языками программирования. Для удобного управления базами данных, на платформу WAMP устанавливается скрипт phpMyAdmin.

Кроме платформ WAMP под Windows, есть платформы MAMP, под Mac ОС, есть платформы LAMP, для Linux. Есть специфические платформы под Windows, называемые WIMP. На них сервер Apache заменяется Internet Information Services (IIS). Разнообразие программных платформ представлены на рисунках 1.2 – 1.6.

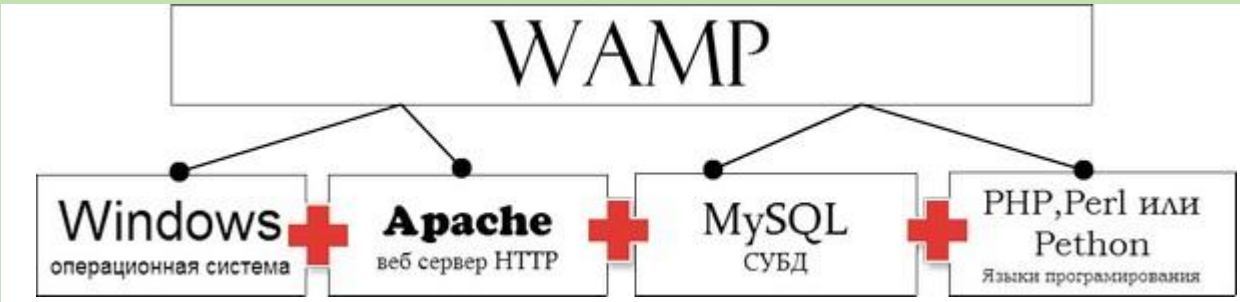


Рисунок 1.2 – Платформа WAMP для Windows

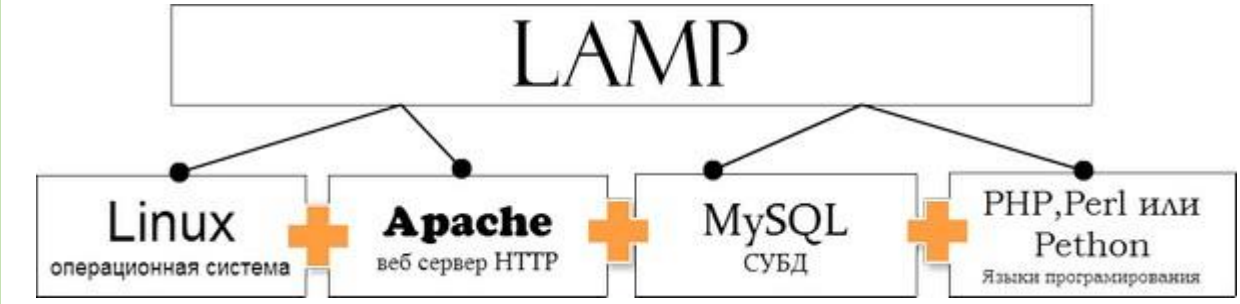


Рисунок 1.3 – Платформа LAMP для Linux

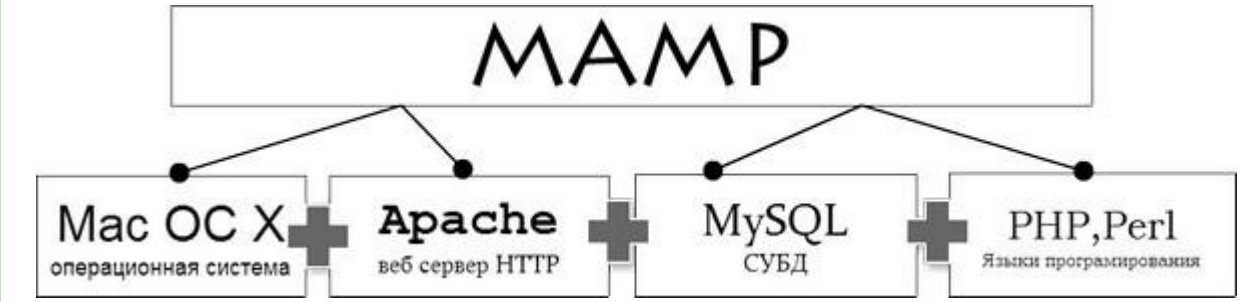


Рисунок 1.4 – Платформа MAMP для Mac OS

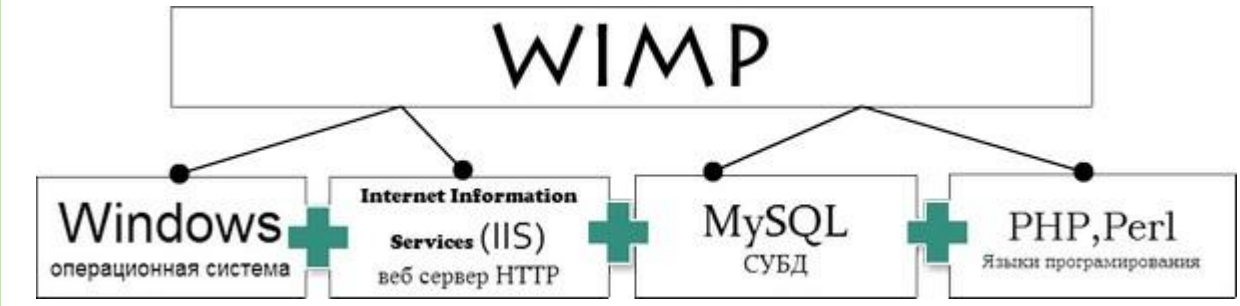


Рисунок 1.5 – Платформа WIMP для Windows с заменой Apache на IIS

Также, работают другие связки, например, вместо Apache ставят сервер Nginx. Такая связка присутствует в платформе OpenServer. Авторы отдельных сборок исключили работу с Apache и полностью перешли на сервер Nginx, например, платформа Winginx.

В то же время есть универсальные платформы, работающие на любой операционной системе, например кроссплатформа XMAPP. XAMPP это акроним от « X- любая ОС, Apache Сервер, MySQL система управления базами данных, PHP язык программирования, Perl язык программирования».

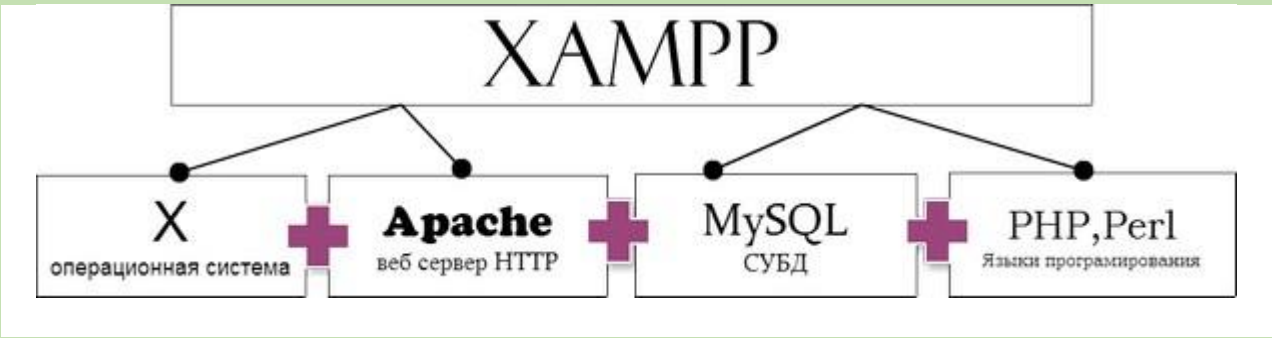


Рисунок 1.6 – Кросслатформа XAMPP

Список популярных платформ WAMP
- кроссплатформа XAMPP;
- руский Денвер;
- мощный Open Server;
- новый WampServer;
- отличный EasyPHP;
- английский VertrigoServ.

	Portable	Русский язык	Наличие GUI	Полезный GUI	Просмотр логов
AMPPS	-	+	+	-	+
AppServ	-	-	-	-	-
Denwer	+	+	-	-	-
EasyPHP	+	+	+	+	+
EleanorServ	+	+	+	+	+
Nimp	+	+	+	+	-
Open Server	+	+	+	+	+
Server2go	+	-	-	-	-
Uniformserver	-	+	+	-	+
USBWebserver	+	-	+	-	-
Vertrigo	-	-	+	+	+
Wampserver	-	+	+	+	+
Winginx	+	+	+	+	+
Xampp USB Lite	+	-	+	-	-

Рисунок 1.7 – Сравнительная таблица основных веб – серверов

2 Разработка и реализация базы данных

2.1 Разработка технического задания

Для создания информационной базы данных гимназии были поставлены нижеперечисленные задачи:

– изучить и проанализировать процесс сбора информации;

– разработать структуру базы данных;

– создать базу данных;

– разработать приложение для наиболее подходящей платформы – Android, обеспечить ограниченный доступ к базе данных из приложения.

2.2 Подготовка к разработке и предустановка программного обеспечения

Пакет для веб – разработки, о составе которого не трудно догадаться из аббревиатуры (Apache, MySQL и PHP). Отличительным преимуществом перед другими пакетами является наличие удобной панели управления (ниже на рисунке).

Сравнение WampServer с другими пакетами:

Почему не Denwer? Сайты должны быть доступны из внешней сети, а денвер позволяет работать с доменами только на локальной машине, можно настроить доступность одного домена через .htaccess, но если доменов больше, то данный пакет не подходит для нашего сервера. Хотя если нужно поднять сервер для какого-то теста, то всегда использую денвер, т.к для его установки нужно нажать три раза enter и не заморачиваться настройками.

WAMP – расшифровывается как связка Apache, MySQL и PHP для Windows, т.е отдельная установка указанных пакетов. Основное преимущество – более гибкая настройка. Основной недостаток – более длительная установка и настройка. Что имеем в итоге: времени будет затрачено раза в 3 больше, а результат получите тот же.

VertrigoServ по сравнению с WampServer имеет расширенный пакет, например, в установщике присутствует пакет Zend Optimizer. Но т.к это заглушка на пару дней, нет смысла поднимать что то сложнее чем WampServer.

После установки WampServer в трее появится его иконка, через которую можно управлять работой и добираться до нужных опций. Например, тем кто не дружит с английским можно подключить русский: ПК на иконке – Language – требуемый язык, или можно запустить файл конфигурации Apache: ЛК на иконке – Apache – httpd.conf.

При попытке зайти на localhost выйдет окно, содержащее основные настройки сервера, возможно в данный момент оно не сильно информативно, т.к у вас ещё не созданы проекты и алиасы, сейчас оно позволяет познакомиться с основными параметрами вашего сервера, представленными на рисунке 2.1.

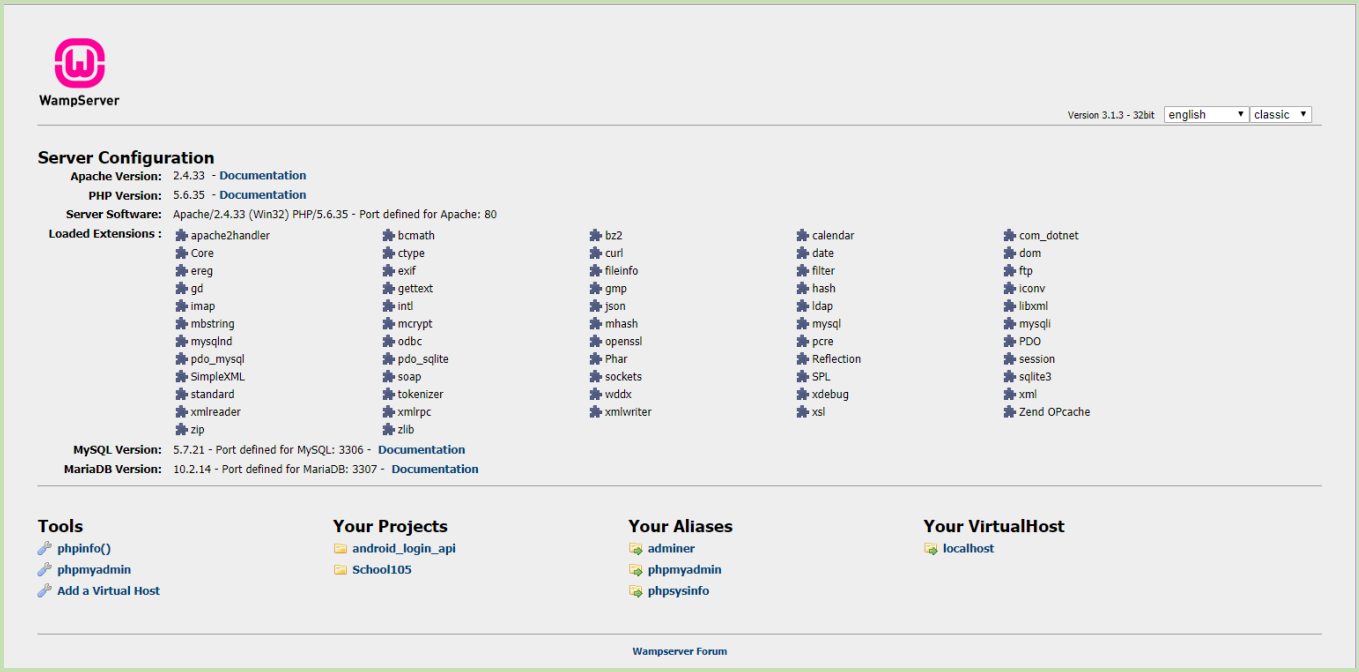


Рисунок 2.1 – Окно, содержащее основные настройки сервера

После установки WAMPServer 3 – переходим к phpmyadmin, проходим процедуру авторизации, где:

Пользователь (по умолчанию) – root;

Пароль – задается при инсталляции WAMPServer 3;

Выбор сервера – MySQL, база данных, в которой и будет храниться основная информация; Окно авторизации можем наблюдать на рисунке 2.2.

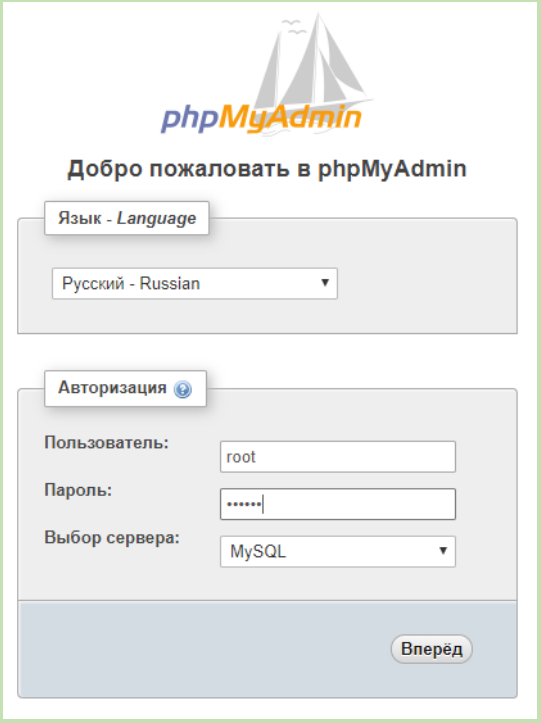


Рисунок 2.2 – Окно авторизации СУБД

После прохождения процедуры авторизации нас встречает главный экран, меню с доступом во все необходимые подпункты и настройки (см рис 2.3)

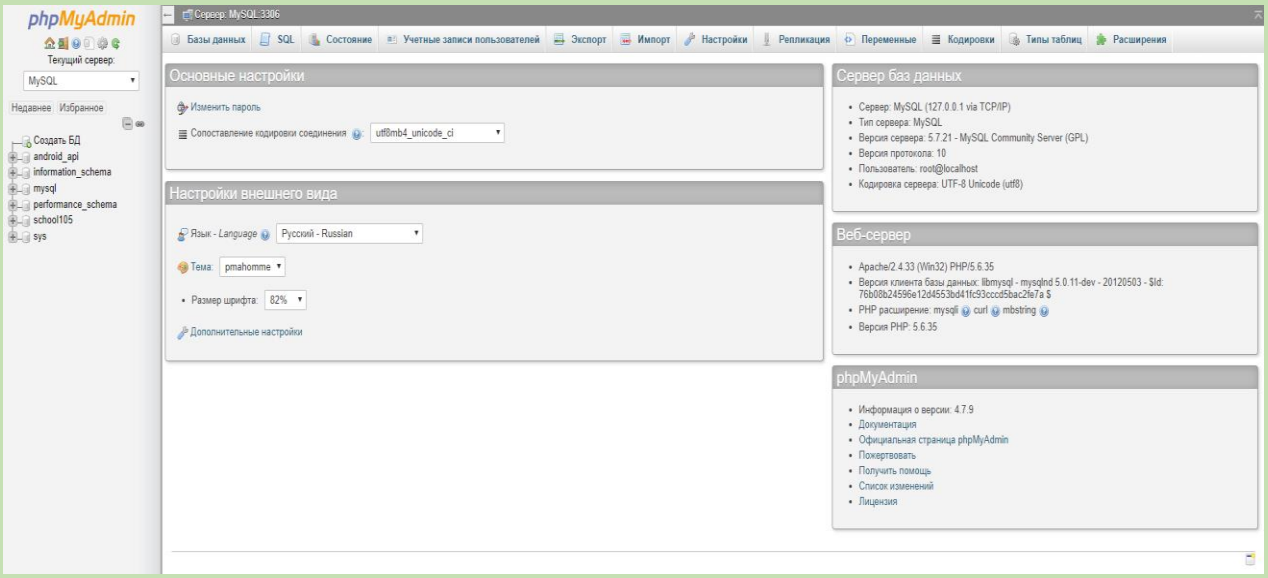


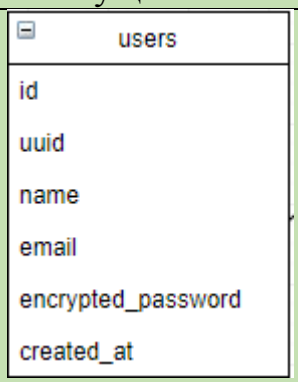
Рисунок 2.3 – Главное окно phpmyadmin

2.3 Разработка структуры базы данных

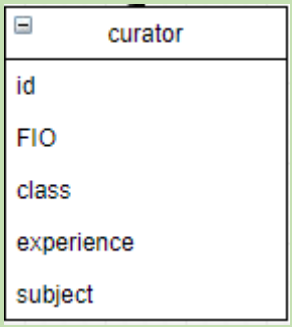
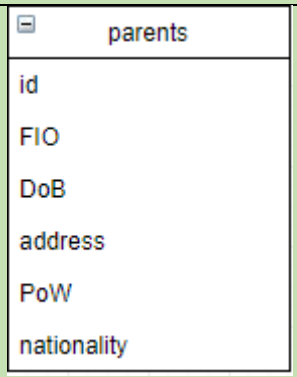
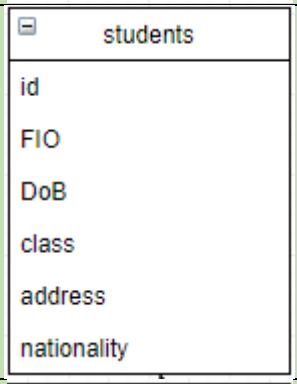
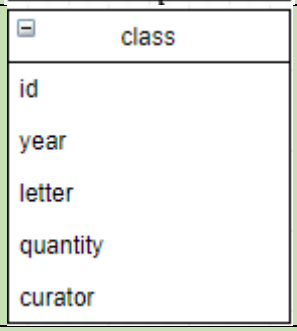
Актуальностью данного дипломного проекта являются: улучшение системы добавления, хранения и поиска информации об учениках и родителях, значительное сокращение времени сбора информации и повышение продуктивности обучения за счет частичной разгрузки преподавателей.

Предметной областью данного проекта является КГУ Гимназия №105. В представленной модели присутствует 5 сущностей (см табл. 2.1 – 2.8): Преподаватель, Ученик, Родитель, Класс, Пользователь. Каждая сущность имеет атрибуты. Между представленными сущностями присутствует связь.

Таблица 2.1 – Описание атрибутов сущностей

Сущности	Описание
	Id - идентификационный номер; Uuid – уникальный идентификационный номер пользователя; Name – полное имя и фамилия; Email – адрес электронной почты; Encrypted_password – зашифрованный пароль; Created_at – дата и время создания записи;

Продолжение таблицы 2.1

	Id – идентификационный номер; FIO – фамилия, имя, отчество; Class – класс, руководство за которым закреплено; Experience – опыт преподавания; Subject – предмет, который ведет преподаватель;
	Id – идентификационный номер; FIO – фамилия, имя, отчество; DoB – дата рождения; Address – адрес проживания; PoW – место работы; Nationality – национальность;
	Id – идентификационный номер; FIO – фамилия, имя, отчество; DoB – дата рождения; Class – класс обучения; Address – адрес проживания; Nationality – национальность;
	Id – идентификационный номер; Year – год обучения; Letter – литера; Quantity – количество учеников в классе; Curator – классный руководитель;

2.4 Составление реляционных отношений

Основой проектирования базы данных является составление реляционных отношений. При составлении реляционных отношений необходимо определить атрибуты и типы данных для каждой сущности в базе данных.

Таблица 2.2 – Структура таблицы пользователей

Column_name	Type_of_data	Null	Example
id	Int(11)	No	7
uuid	Varchar(23)	No	5affdb8fdda362.42192188
Name	Varchar(50)	No	Алексей Ким
Email	Varchar(100)	No	St_lite@mail.ru
Encrypted_password	Varchar(80)	No	dAfpjBPOOTkQsCe9Y
Created_at	datetime	Yes	20.04.2018 15:17:38

Таблица 2.3 – Структура таблицы классных руководителей

Column_name	Type_of_data	Null	Example
Id	Int(10)	No	4
FIO	Varchar(50)	No	Баранова Светлана Николаевна
Class	Varchar(10)	No	6 “Г”
Experience	Int(10)	No	15 лет
Subject	Varchar(20)	No	Математика

Таблица 2.4 – Структура таблицы родителей

Column_name	Type_of_data	Null	Example
Id	Int(10)	No	2
FIO	Varchar(50)	No	Жмурко Евгений Викторович
DoB	Date	No	24.07.1971
Address	Varchar(20)	No	Карасай батыра 84
PoW	Varchar(20)	No	КГУ Лицей №134
Nationality	Varchar(20)	No	Украинец

Таблица 2.5 – Структура таблицы учеников

Column_name	Type_of_data	Null	Example
Id	Int(10)	No	6
FIO	Varchar(50)	No	Евгеньев Анатолий Леонидович
DoB	Date	No	24.11.2003
Class	Varchar(10)	No	8 “Д”
Address	Varchar(50)	No	Гончарова 32
Nationality	Varchar(20)	No	Русский

Таблица 2.6 – Структура таблицы классов

Column_name	Type_of_data	Null	Example
Id	Int(10)	No	10
Year	Int(5)	No	7
Letter	Varchar(5)	No	“А”
Quantity	Int(20)	No	32 ученика
Curator	Varchar(50)	No	Якименко Ирина Масесовна

Таблица 2.7 – Структура таблицы расписание уроков

Column_name	Type_of_data	Null	Example
Day of the week	Varchar(10)	No	Понедельник
Teacher	Varchar(50)	No	Песочина Анна Ивановна
Subject	Varchar(10)	No	География

Таблица 2.8 – Структура таблицы расписание звонков

Column_name	Type_of_data	Null	Example
Shift	Int(5)	No	2

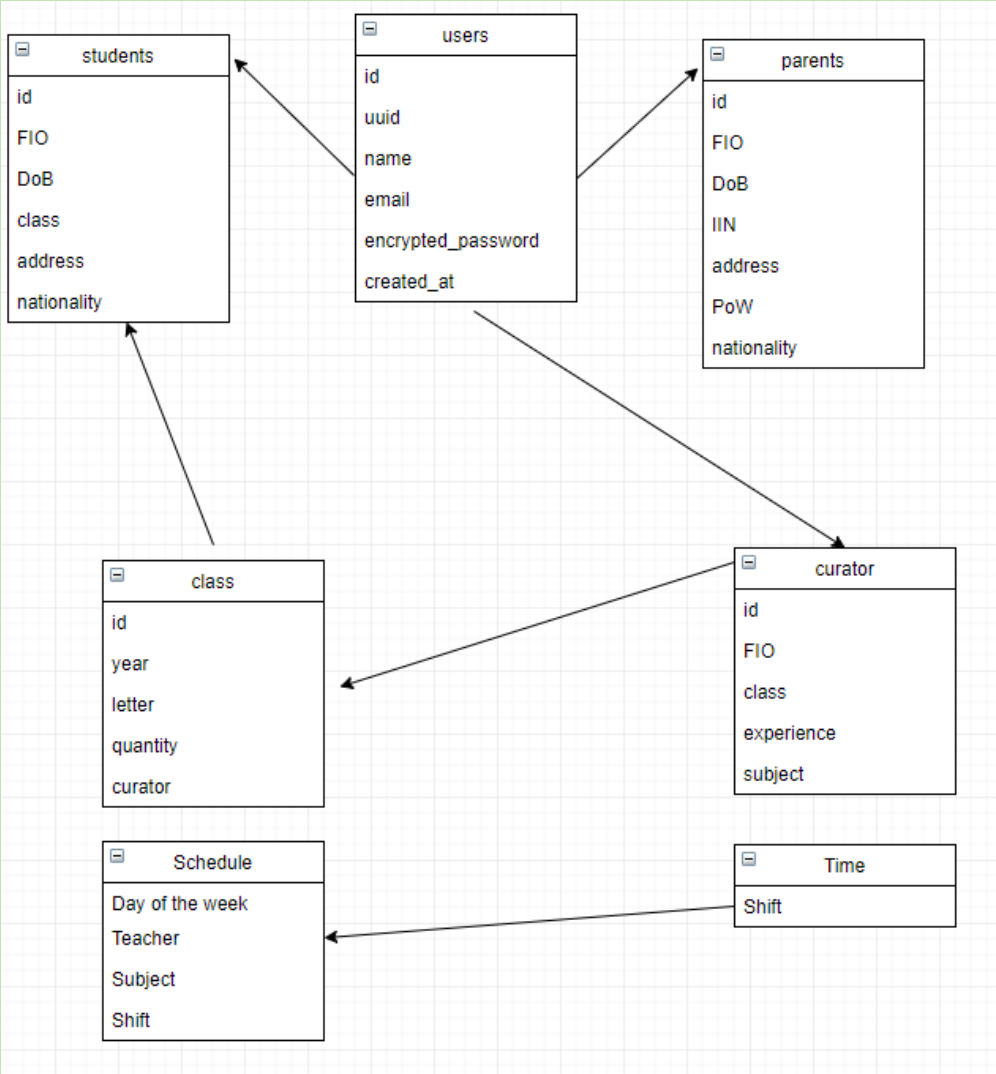


Рисунок 2.4 – Реляционная модель базы данных

2.5 Средства разработки базы данных

База данных – это совокупность связанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования, независимая от прикладных программ. База данных является информационной моделью предметной области. Обращение к базам данных осуществляется с помощью системы управления базами данных (СУБД). СУБД обеспечивает поддержку создания баз данных, централизованного управления и организации доступа к ним различных пользователей.

Итак, мы пришли к выводу, что хранить данные независимо от программ, так, что они связаны между собой и организованы по определенным правилам, целесообразно. Но вопрос, как хранить данные, по каким правилам они должны быть организованы, остался открытым. Способов существует множество

(кстати, называются они моделями представления или хранения данных). Наиболее популярные – объектная и реляционная модели данных.

Автором реляционной модели считается Э. Кодд, который первым предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение) и показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение.

Таким образом, реляционная база данных представляет собой набор таблиц (точно таких же, как приведенная выше), связанных между собой. Строка в таблице соответствует сущности реального мира (в приведенном выше примере это информация о человеке).

Примеры реляционных СУБД: Mysql, PostgreSql. В основу объектной модели положена концепция объектно-ориентированного программирования, в которой данные представляются в виде набора объектов и классов, связанных между собой родственными отношениями, а работа с объектами осуществляется с помощью скрытых (инкапсулированных) в них методов. Примеры объектных СУБД: Cache, GemStone (от Servio Corporation), ONTOS (ONTOS).

В последнее время производители СУБД стремятся соединить два этих подхода и проповедуют объектно-реляционную модель представления данных. Примеры таких СУБД – IBM DB2 for Common Servers, Oracle8.

Поскольку мы собираемся работать с Mysql, то будем обсуждать аспекты работы только с реляционными базами данных. Нам осталось рассмотреть еще два важных понятия из этой области: ключи и индексирование, после чего мы сможем приступить к изучению языка запросов SQL.

Для начала давайте подумаем над таким вопросом: какую информацию нужно дать о человеке, чтобы собеседник точно сказал, что это именно тот человек, сомнений быть не может, второго такого нет? Сообщить фамилию, очевидно, недостаточно, поскольку существуют однофамильцы. Если собеседник человек, то мы можем приблизительно объяснить, о ком речь, например, вспомнить поступок, который совершил тот человек, или еще как-то. Компьютер же такого объяснения не поймет, ему нужны четкие правила, как определить, о ком идет речь. В системах управления базами данных для решения такой задачи ввели понятие первичного ключа.

Первичный ключ (primary key, PK) – минимальный набор полей, уникально идентифицирующий запись в таблице. Значит, первичный ключ – это в первую очередь набор полей таблицы, во-вторых, каждый набор значений этих полей должен определять единственную запись (строку) в таблице и, в третьих, этот набор полей должен быть минимальным из всех обладающих таким же свойством.

Поскольку первичный ключ определяет только одну уникальную запись, то никакие две записи таблицы не могут иметь одинаковых значений первичного ключа.

Например, в нашей таблице (см. выше) ФИО и адрес позволяют однозначно выделить запись о человеке. Если же говорить в общем, без связи с решаемой задачей, то такие знания не позволяют точно указать на единственного человека, поскольку существуют однофамильцы, живущие в разных городах по одному адресу. Все дело в границах, которые мы сами себе задаем. Если считаем, что знания ФИО, телефона и адреса без указания города для наших целей достаточно, то все замечательно, тогда поля ФИО и адрес могут образовывать первичный ключ. В любом случае проблема создания первичного ключа ложится на плечи того, кто проектирует базу данных (разрабатывает структуру хранения данных). Решением этой проблемы может стать либо выделение характеристик, которые естественным образом определяют запись в таблице (задание так называемого логического, или естественного, PK), либо создание дополнительного поля, предназначенного именно для однозначной идентификации записей в таблице (задание так называемого суррогатного, или искусственного, PK).

Примером логического первичного ключа является номер паспорта в базе данных о паспортных данных жителей или ФИО и адрес в телефонной книге (таблица выше). Для задания суррогатного первичного ключа в нашу таблицу можно добавить поле id (идентификатор), значением которого будет целое число, уникальное для каждой строки таблицы. Использование таких суррогатных ключей имеет смысл, если естественный первичный ключ представляет собой большой набор полей или его выделение нетривиально.

Кроме однозначной идентификации записи, первичные ключи используются для организации связей с другими таблицами. Например, у нас есть три таблицы: содержащая информацию об исторических личностях (Persons), содержащая информацию об их изобретениях (Artifacts) и содержащая изображения как личностей, так и артефактов (Images) (рис 10.1).

Первичным ключом во всех этих таблицах является поле id (идентификатор). В таблице Artifacts есть поле author, в котором записан идентификатор, присвоенный автору изобретения в таблице Persons. Каждое значение этого поля является внешним ключом для первичного ключа таблицы Persons. Кроме того, в таблицах Persons и Artifacts есть поле photo, которое ссылается на изображение в таблице Images. Эти поля также являются внешними ключами для первичного ключа таблицы Images и устанавливают однозначную логическую связь Persons-Images и Artifacts-Images. То есть если значение внешнего ключа photo в таблице личности равно 10, то это значит, что фотография этой личности имеет id=10 в таблице изображений. Таким образом, внешние ключи используются для организации связей между таблицами базы данных (родительскими и дочерними) и для поддержания ограничений ссылочной целостности данных.

Одна из основных задач, возникающих при работе с базами данных, – это задача поиска. При этом, поскольку информации в базе данных, как правило, содержится много, перед программистами встает задача не просто поиска, а

эффективного поиска, т.е. поиска за сравнительно небольшое время и с достаточной точностью.

Для этого (для оптимизации производительности запросов) производят индексирование некоторых полей таблицы. Использовать индексы полезно для быстрого поиска строк с указанным значением одного столбца. Без индекса чтение таблицы осуществляется по всей таблице, начиная с первой записи, пока не будут найдены соответствующие строки. Чем больше таблица, тем больше накладные расходы. Если же таблица содержит индекс по рассматриваемым столбцам, то база данных может быстро определить позицию для поиска в середине файла данных без просмотра всех данных. Это происходит потому, что база данных помещает проиндексированные поля поближе в памяти, так, чтобы можно было побыстрее найти их значения. Для таблицы, содержащей 1000 строк, это будет как минимум в 100 раз быстрее по сравнению с последовательным перебором всех записей. Однако в случае, когда необходим доступ почти ко всем 1000 строкам, быстрее будет последовательное чтение, так как при этом не требуется операций поиска по диску. Так что иногда индексы бывают только помехой. Например, если копируется большой объем данных в таблицу, то лучше не иметь никаких индексов. Однако в некоторых случаях требуется задействовать сразу несколько индексов (например, для обработки запросов к часто используемым таблицам).

Если говорить о Mysql, то там существует три вида индексов: PRIMARY, UNIQUE, и INDEX, а слово ключ (KEY) используется как синоним слова индекс (INDEX). Все индексы хранятся в памяти в виде B-деревьев.

PRIMARY – уникальный индекс (ключ) с ограничением, что все индексированные им поля не могут иметь пустого значения (т.е. они NOT NULL). Таблица может иметь только один первичный индекс, но он может состоять из нескольких полей.

UNIQUE – ключ (индекс), задающий поля, которые могут иметь только уникальные значения.

INDEX – обычный индекс (как мы описали выше). В Mysql, кроме того, можно индексировать строковые поля по заданному числу символов от начала строки.

Mysql – это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. Mysql – свободно распространяемая система, т.е. платить за ее применение не нужно. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД, вполне подходящая для не слишком глобальных проектов.

Работать с Mysql можно не только в текстовом режиме, но и в графическом. Существует очень популярный визуальный интерфейс (кстати, написанный на PHP) для работы с этой СУБД. Называется он PhpMyAdmin. Этот интерфейс позволяет значительно упростить работу с базами данных в Mysql.

PhpMyAdmin позволяет пользоваться всеми достоинствами браузера, включая прокрутку изображения, если оно не умещается на экран. Многие из базовых SQLфункций работы с данными в PhpMyAdmin сведены к интуитивно понятным интерфейсам и действиям, напоминающим переход по ссылкам в Internet. Но, тем не менее, стоит все же поработать и в текстовом режиме.

Перед тем как переходить к детальному изучению языка SQL, несколько слов об установке Mysql и подготовке к работе. Если вы не собираетесь заниматься администрированием сервера, то информация, приведенная ниже, пригодится вам только для общего развития. Итак, устанавливается Mysql очень просто – автоматически, пару раз нажмите OK, и все. После этого вы можете зайти в директорию, где лежат файлы типа Mysql.exe, Mysqld.exe и т.п. (у нас под Windows XP это c:\Mysql\bin) Последний файл запускает Mysql-сервер. В некоторых системах сервер запускается в виде сервиса. После запуска сервера следует запустить Mysql-клиент, запустив программу Mysql.exe. Здесь даже пароля не спросят. Более того, если вы наберете shell> Mysql.exe -u root или shell>Mysql -u root Mysql то получите все права администратора Mysql сервера. Кстати, выполнять эти команды надо, находясь в той директории, где лежат файлы Mysql.exe.

Первый международный стандарт языка SQL был принят в 1989 г., его часто называют SQL/89. Среди недостатков этого стандарта выделяют в первую очередь то, что многие важные свойства он устанавливал, как определяемые в реализации.

Отсюда произошло множество расхождений в реализациях языка разными производителями. Кроме того, высказывались претензии по поводу отсутствия в этом стандарте упоминаний о практических аспектах языка, таких как его встраивание в язык программирования Си.

Следующий международный стандарт языка SQL был принят в конце 1992 г. И стал называться SQL/92. Он получился гораздо более точным и полным, чем SQL/89, хотя и не был лишен недостатков. В настоящее время большинство систем почти полностью реализуют этот стандарт. Однако, как известно, прогресс не остановишь, и в 1999 году появился новый стандарт SQL:1999, также известный как SQL3. SQL3 характеризуется как «объектно-ориентированный SQL» и является основой нескольких объектно-реляционных систем управления базамиданных (например, ORACLE8 компании Oracle, Universal Server компании Informix и DB2 Universal Database компании IBM). Этот стандарт является не просто слияниемSQL-92 и объектнойтехнологии. Он содержит ряд расширений традиционного SQL, а сам документ составлен таким образом, чтобы добиться более эффективной работы в области стандартизации в будущем.

Если говорить о Mysql, то она соответствует начальному уровню SQL92, содержит несколько расширений этого стандарта и стремится к полной поддержке стандарта ANSI SQL99, но без ущерба для скорости и качества кода.

2.6 Разработка связи с базой данных

Для полного взаимодействия с базой данных MySQL, в первую очередь необходимо создать REST API (Application Programming Interface) – Интерфейс прикладного программирования. При создании приложения для платформы Android, которое в последующем будет управлять всеми пользовательскими данными в центральной базе данных, REST API является хорошей архитектурной опцией для обмена данными между приложением и сервером. REST представляет репрезентативный перенос состояний. Реализация REST очень проста по сравнению с другими методами, такими как SOAP, CORBA, WSDL и т.д. В основном работает по протоколу HTTP.

Задача REST Api - получить запрос от клиента, взаимодействовать с базой данных и, наконец, дать ответ клиенту. Поэтому мы сначала создадим простой PHP, MySQL API. Наш API выполняет следующие задачи:

– Принимает запросы в методах GET/POST;

– Взаимодействует с базой данных посредством вставки/выборки данных;

– Возвращает ответ в формате JSON.

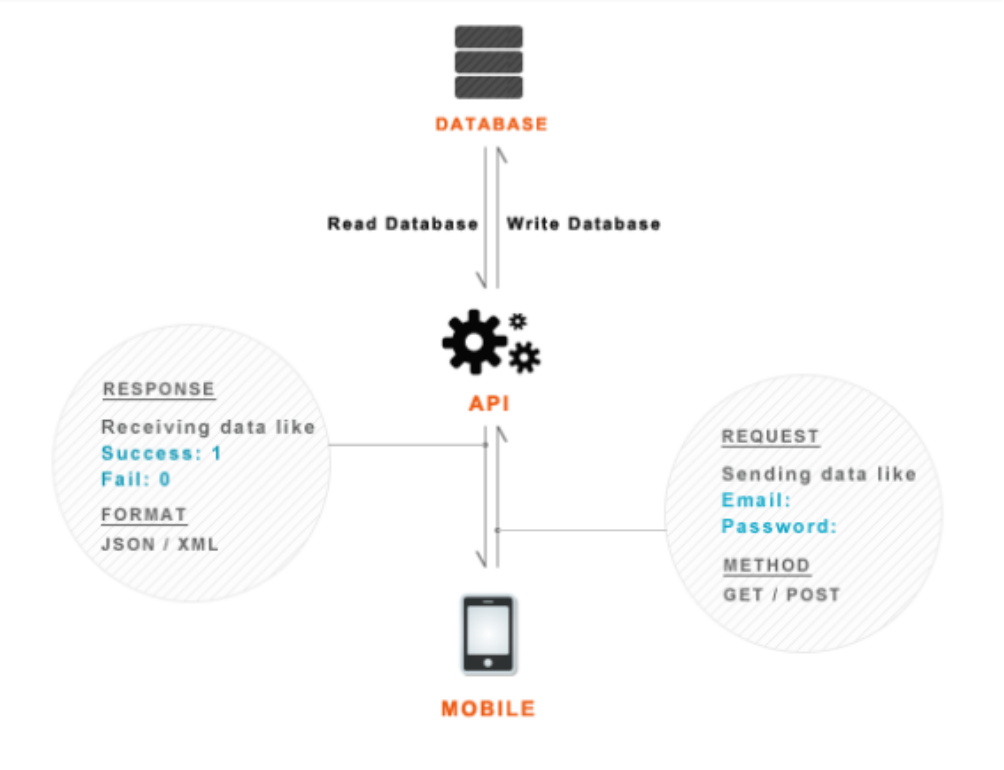


Рисунок 2.6 – соединение Android с PHP и MySQL

Связь между приложением и базой данных происходит посредством PHP. (см. рис. 2.6). Для корректной работы приложения с базой необходимо создать несколько файлов. Для этого в папке с локальным сервером, далее – www, создается папка с названием проекта. В папке с проектом создается еще одна папка – include. И далее создается файл – Config.php: (см. рис. 2.7)

Config.php

```php
<?php

/**
 * Database config variables
 */
define("DB_HOST", "localhost");
define("DB_USER", "root");
define("DB_PASSWORD", "");
define("DB_DATABASE", "android_api");
?>
```

Рисунок 2.7 – Файл, содержащий имя пользователя и пароль MySQL

Далее создаем класс, который отвечает за открытие и закрытие соединения с базой данных – DB_Connect.php: (см. рис. 2.8).

DB_Connect.php

```php
<?php
class DB_Connect {
    private $conn;

    // Connecting to database
    public function connect() {
        require_once 'include/Config.php';

        // Connecting to mysql database
        $this->conn = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_DATABASE);

        // return database handler
        return $this->conn;
    }
}

?>
```

Рисунок 2.8 – содержание файла открытия соединения с БД

Далее создается файл с функцией хранения пользователя, и получение пользователя из базы данных: (см. рис. 2.9, 2.10).

DB_Functions.php

```php
<?php

class DB_Functions {

    private $conn;

    // constructor
    function __construct() {
        require_once 'DB_Connect.php';
        // connecting to database
        $db = new Db_Connect();
        $this->conn = $db->connect();
    }

    // destructor
    function __destruct() {

    }

    /**
     * Storing new user
     * returns user details
     */
    public function storeUser($name, $email, $password) {
        $uuid = uniqid('', true);
        $hash = $this->hashSSHA($password);
        $encrypted_password = $hash["encrypted"]; // encrypted password
        $salt = $hash["salt"]; // salt

        $stmt = $this->conn->prepare("INSERT INTO users(unique_id, name, email, encrypted_password, salt, created_at) VALUES(?, ?, ?, ?, ?, NOW())");
        $stmt->bind_param("sssss", $uuid, $name, $email, $encrypted_password, $salt);
        $result = $stmt->execute();
        $stmt->close();

        // check for successful store
        if ($result) {
            $stmt = $this->conn->prepare("SELECT * FROM users WHERE email = ?");
            $stmt->bind_param("s", $email);
            $stmt->execute();
            $user = $stmt->get_result()->fetch_assoc();
            $stmt->close();

            return $user;
        } else {
            return false;
        }
    }
```

Рисунок 2.9 – Фрагмент файла с функцией хранения пользователя в БД

DB_Functions.php

```php
    public function addinfo($uuid, $name, $DoB, $address, $PoW, $child) {

        $stmt = $this->conn->prepare("INSERT INTO info(uuid, name, DoB, address, PoW, Child) VALUES(?, ?, ?, ?, ?, ?)");
        $stmt->bind_param("ssssss", $uuid, $name, $DoB, $address, $PoW, $child);
        $result = $stmt->execute();
        $stmt->close();

        // check for successful store
        if ($result) {
                        $stmt = $this->conn->prepare("SELECT * FROM info WHERE uuid = ?");
            $stmt->bind_param("s", $uuid);
            $stmt->execute();
            $user = $stmt->get_result()->fetch_assoc();
            $stmt->close();

            return $user;
        } else {
            return "error";
        }
    }


    /**
     * Get user by email and password
     */
    public function getUserByEmailAndPassword($email, $password) {

        $stmt = $this->conn->prepare("SELECT * FROM users WHERE email = ?");

        $stmt->bind_param("s", $email);

        if ($stmt->execute()) {
            $user = $stmt->get_result()->fetch_assoc();
            $stmt->close();

            // verifying user password
            $salt = $user['salt'];
            $encrypted_password = $user['encrypted_password'];
            $hash = $this->checkhashSSHA($salt, $password);
            // check for password equality
            if ($encrypted_password == $hash) {
                // user authentication details are correct
                return $user;
            }
        } else {
            return NULL;
        }
    }
```

Рисунок 2.10 – Фрагмент файла с функцией получения пользователя и добавление информации о нем в БД

После создаем эндпоинт регистрации (2.11), который принимает значения имени, электронной почты и пароль как POST параметры и хранит данные пользователя в базе данных MySQL:

register.php

```php
<?php

require_once 'include/DB_Functions.php';
$db = new DB_Functions();

// json response array
$response = array("error" => FALSE);

if (isset($_POST['name']) && isset($_POST['email']) && isset($_POST['password'])) {

    // receiving the post params
    $name = $_POST['name'];
    $email = $_POST['email'];
    $password = $_POST['password'];

    // check if user is already existed with the same email
    if ($db->isUserExisted($email)) {
        // user already existed
        $response["error"] = TRUE;
        $response["error_msg"] = "User already existed with " . $email;
        echo json_encode($response);
    } else {
        // create a new user
        $user = $db->storeUser($name, $email, $password);
        if ($user) {
            // user stored successfully
            $response["error"] = FALSE;
            $response["uid"] = $user["unique_id"];
            $response["user"]["name"] = $user["name"];
            $response["user"]["email"] = $user["email"];
            $response["user"]["created_at"] = $user["created_at"];
            $response["user"]["updated_at"] = $user["updated_at"];
            echo json_encode($response);
        } else {
            // user failed to store
            $response["error"] = TRUE;
            $response["error_msg"] = "Unknown error occurred in registration!";
            echo json_encode($response);
        }
    }
} else {
    $response["error"] = TRUE;
    $response["error_msg"] = "Required parameters (name, email or password) is missing!";
    echo json_encode($response);
}
?>
```

Рисунок 2.11 – Листинг регистрационного эндпоинта

По вышеуказанному примеру создаем эндпоинт для авторизации (рис. 2.12), который принимает электронную почту и пароль как POST параметры. После получения электронной почты и пароля данный эндпоинт проверяет на совпадения в БД. А после создаем файл с добавлением информации в БД (рис. 2.13).

```php
<?php
require_once 'include/DB_Functions.php';
$db = new DB_Functions();

// json response array
$response = array("error" => FALSE);

if (isset($_POST['email']) && isset($_POST['password'])) {

    // receiving the post params
    $email = $_POST['email'];
    $password = $_POST['password'];

    // get the user by email and password
    $user = $db->getUserByEmailAndPassword($email, $password);

    if ($user != false) {
        // use is found
        $response["error"] = FALSE;
        $response["uid"] = $user["unique_id"];
        $response["user"]["name"] = $user["name"];
        $response["user"]["email"] = $user["email"];
        $response["user"]["created_at"] = $user["created_at"];
        $response["user"]["updated_at"] = $user["updated_at"];
        echo json_encode($response);
    } else {
        // user is not found with the credentials
        $response["error"] = TRUE;
        $response["error_msg"] = "Login credentials are wrong. Please try again!";
        echo json_encode($response);
    }
} else {
    // required post params is missing
    $response["error"] = TRUE;
    $response["error_msg"] = "Required parameters email or password is missing!";
    echo json_encode($response);
}
?>
```

Рисунок 2.12 – Листинг авторизационного эндпоинта

```php
<?php

require_once 'include/DB_Functions.php';
$db = new DB_Functions();

// json response array
$response = array("error" => FALSE);

if (isset($_POST['uuid']) && isset($_POST['name']) && isset($_POST['DoB']) && isset($_POST['address']) && isset($_POST['PoW']) && isset($_POST['child'])) {

    // receiving the post params
    $uuid = $_POST['uuid'];
    $name = $_POST['name'];
    $DoB = $_POST['DoB'];
    $address = $_POST['address'];
    $PoW = $_POST['PoW'];
    $child = $_POST['child'];


        // create a new user
        $user = $db->addinfo($uuid, $name, $DoB, $address, $PoW, $child);

} else {
    $response["error"] = TRUE;
    $response["error_msg"] = "Required parameters (name, email or password) is missing!";
    echo json_encode($response);
}
?>
```

Рисунок 2.13 – Листинг файла с добавлением информации

3 Разработка пользовательского графического интерфейса приложения

3.1 Установка Android Studio, первичная настройка и установка необходимых компонентов для полного функционирования среды.

Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее.

Язык Java зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов. Реализация проекта была начата на языке C++, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента — языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

Язык Java потребовался для создания интерактивных продуктов для сети Internet. Фактически, большинство архитектурных решений, принятых при создании Java, было продиктовано желанием предоставить синтаксис, сходный с C и C++. В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты C++.

Три ключевых элемента объединились в технологии языка Java:

– java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML;

– java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты;

– java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе

с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

Java является основой практически для всех типов сетевых приложений и всеобщим стандартом для разработки и распространения встроенных и мобильных приложений, игр, веб-контента и корпоративного программного обеспечения. В мире насчитывается более 9 миллионов специалистов, разрабатывающих приложения на Java, которая позволяет эффективно разрабатывать, внедрять, использовать превосходные приложения и услуги.

От портативных компьютеров до центров сбора данных, от игровых консолей до суперкомпьютеров, используемых для научных разработок, от сотовых телефонов до сети Интернет — Java используется повсюду:

– java используется на 97% корпоративных настольных ПК;

– java используется на 89% настольных ПК в США;

– 9 млн разработчиков на Java в мире;

– инструмент номер 1 среди разработчиков;

– программа номер 1 среди разработчиков;

– java используется в 3 млрд мобильных телефонов;

– java входит в комплект поставки 100% всех проигрывателей дисков Blu-ray;

– используется 5 млн Java Card;

– java используется в 125 млн ТВ-устройств;

– 5 из 5 основных производителей оригинального оборудования включают в комплект поставки Java ME.

Технология Java протестирована, усовершенствована, расширена и проверена участниками сообщества разработчиков Java, архитекторов и энтузиастов.Java позволяет разрабатывать производительные, портативные приложения практически на всех компьютерных платформах. Доступность приложений в разнородных средах позволяет компаниям предоставлять более широкий спектр услуг, способствует повышению производительности, уровня взаимодействия и совместной работы конечных пользователей и существенному снижению стоимости совместного владения корпоративными и потребительскими приложениями. Java стала незаменимым инструментом для разработчиков и открыла для них следующие возможности:

– написание программного обеспечения на одной платформе и его запуск практически на любой другой платформе;

– создание программ, работающих в веб-браузере и имеющих доступ к веб-службам;

– разработка приложений на стороне сервера для форумов в Интернете, магазинов, опросов, обработки форм HTML и много другого;

– объединение приложений или служб с использованием языка Java для создания высокоспециализированных приложений или служб;

– создание многофункциональных и эффективных приложений для мобильныхтелефонов, удаленных процессоров, микроконтроллеров, беспроводных модулей, датчиков, шлюзов, потребительских продуктов и практически любых других категорий электронных устройств.

Во многих колледжах и университетах преподаются курсы по программированию на платформе Java. Академия Oracle предоставляет учреждениям школьного, профессионального и высшего образования полный портфель программного обеспечения, учебные курсы, хостинговые технологии, факультативное обучение, поддержку и ресурсы сертификации для использования в учебных целях, а также поддержку Java для сотен тысяч студентов. Разработчики также могут повысить свою квалификацию в области программирования на Java с помощью материалов, доступных на веб-сайте Oracle для разработчиков Java, подписки на информационные рассылки, посвященные технологии Java, и журнал Java Magazine, использования учебных пособий по Java и центров программирования для начинающих разработчиков Java, а также участия в веб-, виртуальных или проводимых инструкторами учебных курсах и сертификациях.

Сеть Oracle Technology Network - самое большое в мире сообщество разработчиков приложений, администраторов баз данных, системных администраторов/разработчиков и архитекторов, использующих стандартные технологии в сочетании с продуктами Oracle. Это сообщество также содержит информационную базу java.oracle.com, наиболее полный официальный источник технической информации о Java. Членство в сообществе бесплатно, присоединяйтесь сегодня! (В разделе 'Участие в сообществах' вашего профиля установите флажок 'Oracle Technology Network'.)

Юные разработчики начинают изучать языки программирования с самого раннего возраста. С помощью визуальных инструментов обучения, таких как Alice, Greenfoot и BlueJ, новое поколение может научиться программировать на языке Java и на легких в использовании языках программирования, разработанных на основе Java.

JavaFX работает на основе Java. На платформе JavaFX разработчики могут создавать и развертывать полнофункциональные интернет-приложения (RIA), одинаково стабильно функционирующие на различных платформах. JavaFX расширяет возможности Java, позволяя разработчикам использовать любые библиотеки Java в приложениях JavaFX. Разработчики могут расширить свои возможности в Java и воспользоваться технологией презентаций, обеспечиваемой JavaFX для создания увлекательных визуальных образов.

Первая характеристика,объектно-ориентированный подход ("ОО"), относится к методу программирования для чайников и дизайну язык. Основная идея ОО заключается в разработке программного обеспечения все это "вещи" (то есть объекты), которыми можно манипулировать, а не действия, которые нужно выполнять. Это основано на том, что первое (вещи) изменяются реже и радикальнее, чем действия, что делает такие объекты (на самом деле вещи, содержащие данные) более стабильной основу для разработки программного

обеспечения. Целью является, делать большие проекты программного обеспечения легко управляемыми, таким образом, можно добиться повышения качества и сокращения числа неудачных проектов программирования для начинающих.

Вторая характеристика, независимость от платформы, означает, что программы, написанные на языке Java должны работать аналогично на различном оборудовании. Программист должен быть в состоянии написать программу один раз и запустить её в любом месте. Это достигается путем компиляции Java-кода "наполовину" в байт-код - упрощенные машинные команды, которым соответствуют стандартный набор реальных команд процессору. Этот код затем необходимо запустить на виртуальной машине, то есть программе, написаной на машинном коде для взаимодействия с аппаратными средствами, которая переводит байт-код Java в пригодный для использования код на конкретном оборудовании. Кроме того, предоставляются стандартизированные библиотеки для обеспечения доступа к особенностям архитектуры конкретной машины (например, графики и сетей) единым способам. Язык Java также включает поддержку для многопоточных программ - жизненно важная необходимость для многих сетевых приложений и основа программирования.

Первая реализация языка использовали интерпретирующую виртуальную машину для достижения мобильности, и многие реализации используют такой подход до сих пор. Однако, эти реализации создают программы, которые выполняются медленнее, чем полностью скомпилированные программы, созданные типичным компилятором C++ и немного позднее появившимися компиляторами языка Java, поэтому язык страдает от приобретенной репутации "производитель медленных программ". В более поздних реализациях Java VM создает программы, которые работают намного быстрее, используя несколько методов.

Первый метод – это просто компиляция непосредственно в машинный код, как традиционные компилятор, пропуская этап превращения программы в байт-код целиком. Этим достигается большая производительность, но за счет утраты мобильности и переносимости программ. Другой метод, "В процессе исполнения" или "JIT", компилирует байт-код Java в машинный код во время выполнения программы. Более сложные виртуальные машины даже использовали динамические перекомпиляции, в которой VJM может анализировать поведение работы программы и выборочно перекомпилировать, и оптимизировать критические части программы. Обе эти технологии позволяют программе воспользоваться скоростью машинного код без потери мобильности.

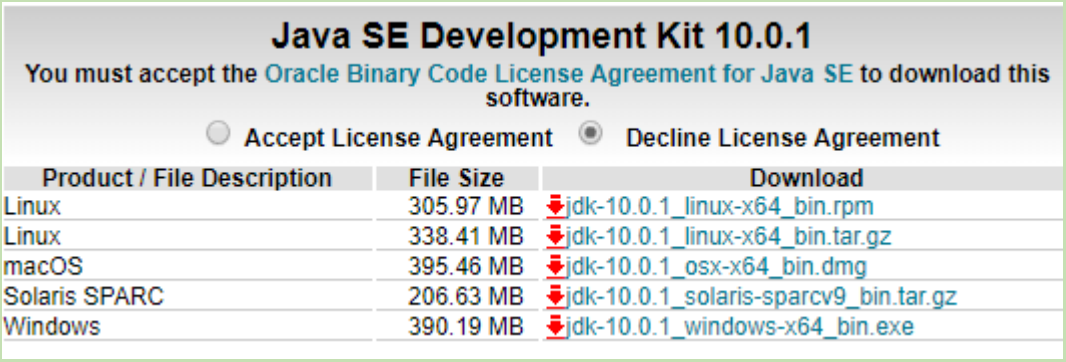
Портативность является технически трудной целью для достижения и успех Java в достижении этой цели является предметом споров. Хотя на самом деле можно писать программы для платформы Java, которые ведут себя одинаково на многих платформах, но все же большее количество доступных платформ не выполняет программу как ожидалось, а выдает небольшое

количество ошибок или несоответствий, что привело к появлению пародии на известный лозунг компании Sun "Написал один раз, работает везде" в другой "Написал один раз, отлаживал везде ".

Независимый от платформы Java, однако, стал очень успешным для серверов приложений, таких как веб-сервисы, сервлеты, или Enterprise Java Beans.

Платформа Java была одной из первых систем для обеспечения широкой поддержки для выполнения кода из удаленных источников. Апплет может работать в браузере пользователя, в процессе выполнения кода может загрузить маленький кусочек чужого кода с удаленного сервера HTTP и выполнить. Удаленное выполнения кода происходит в весьма ограниченной "песочнице", которая защищает пользователя от некорректного или вредоносного кода. Издатели таких приложений могут подать заявку на сертификат, который они могли бы использовать для цифровой подписи апплетов как "безопасный", что дает им разрешение, чтобы вырваться из "песочницы" и получить доступ к локальной файловой системе и сети, конечно, предположительно этот процесс происходит под контролем пользователя.

Для начала скачиваем актуальную версию JDK (Java development kit), с официального сайта, совместимую с вашей операционной системой, в нашем случае – это Windows (рис. 3.1).



Java SE Development Kit 10.0.1

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux	305.97 MB	jdk-10.0.1_linux-x64_bin.rpm
Linux	338.41 MB	jdk-10.0.1_linux-x64_bin.tar.gz
macOS	395.46 MB	jdk-10.0.1_osx-x64_bin.dmg
Solaris SPARC	206.63 MB	jdk-10.0.1_solaris-sparcv9_bin.tar.gz
Windows	390.19 MB	jdk-10.0.1_windows-x64_bin.exe

Рисунок 3.1 – выбор совместимого пакета JDK

Одним из фундаментальных шагов является добавление в переменные среды данного пакета для полноценного функционирования в системе(рис. 3.2):

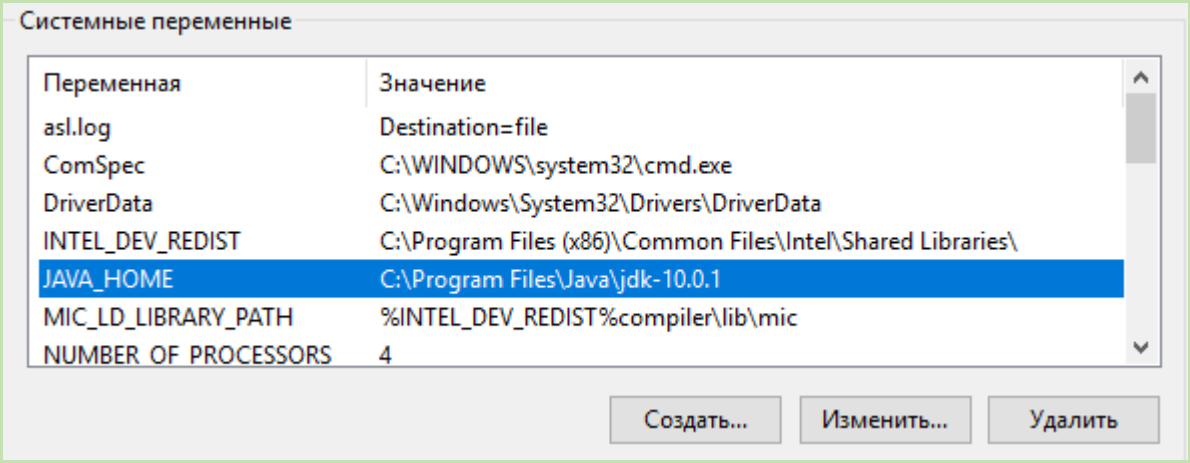


Рисунок 3.2 – Добавление системной переменной JAVA_HOME

Далее осуществляем проверку корректной установки JDK на наш рабочий ноутбук посредством вызова командной строки (рис. 3.3):

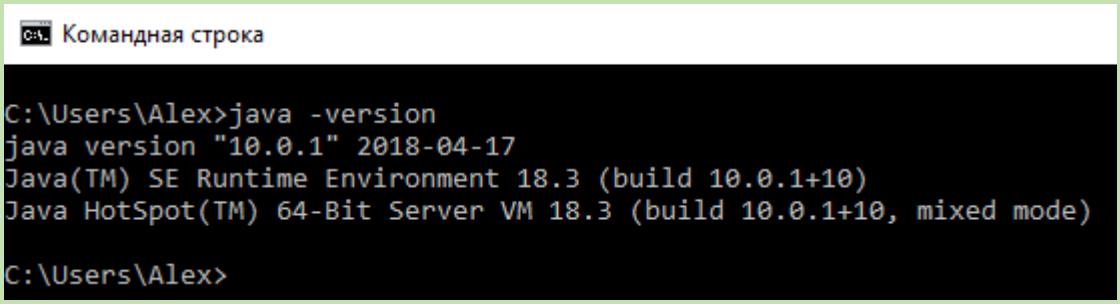


Рисунок 3.3 – Результат запроса установленной версии Java

Приступаем к установке интегрированной среде разработки – Android Studio. Вместе с Android Studio (рис. 3.4, 3.5) скачивается и необходимый SDK (software development kit), что несомненно является одним из плюсов Android studio. Еще одним преимуществом является наличие встроенного эмулятора андроид устройств.

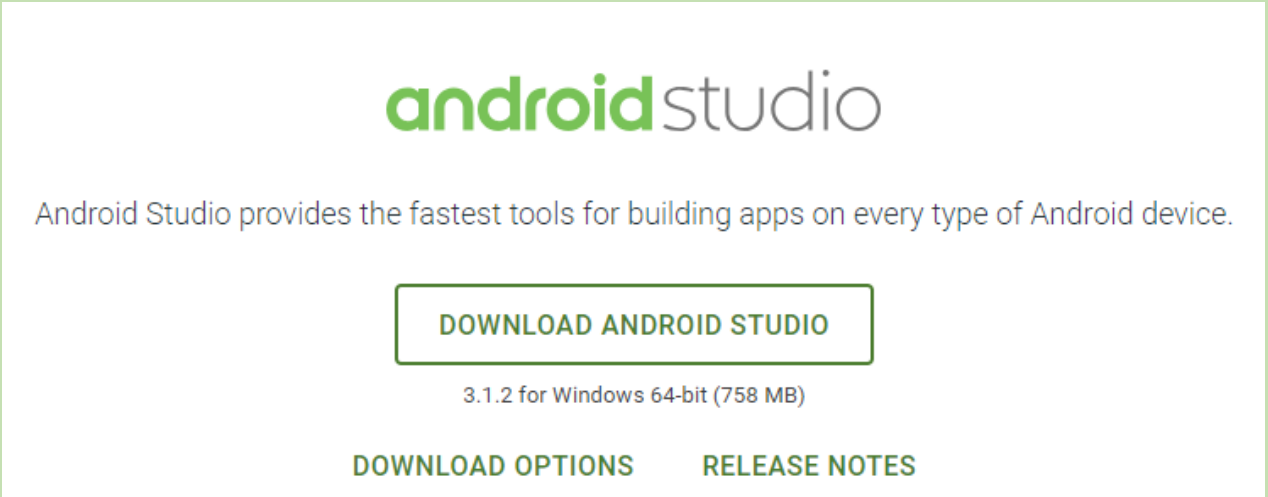


Рисунок 3.4 – Скачивание необходимых компонентов Android studio

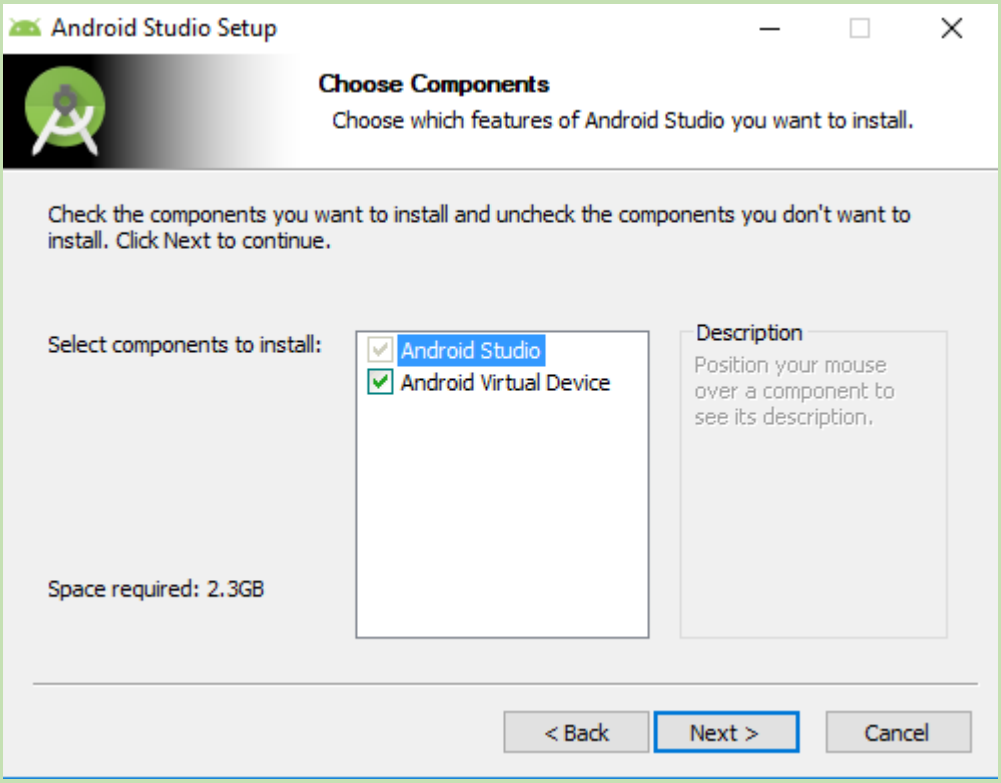


Рисунок 3.5 – Окно установки Android studio и SDK пакета

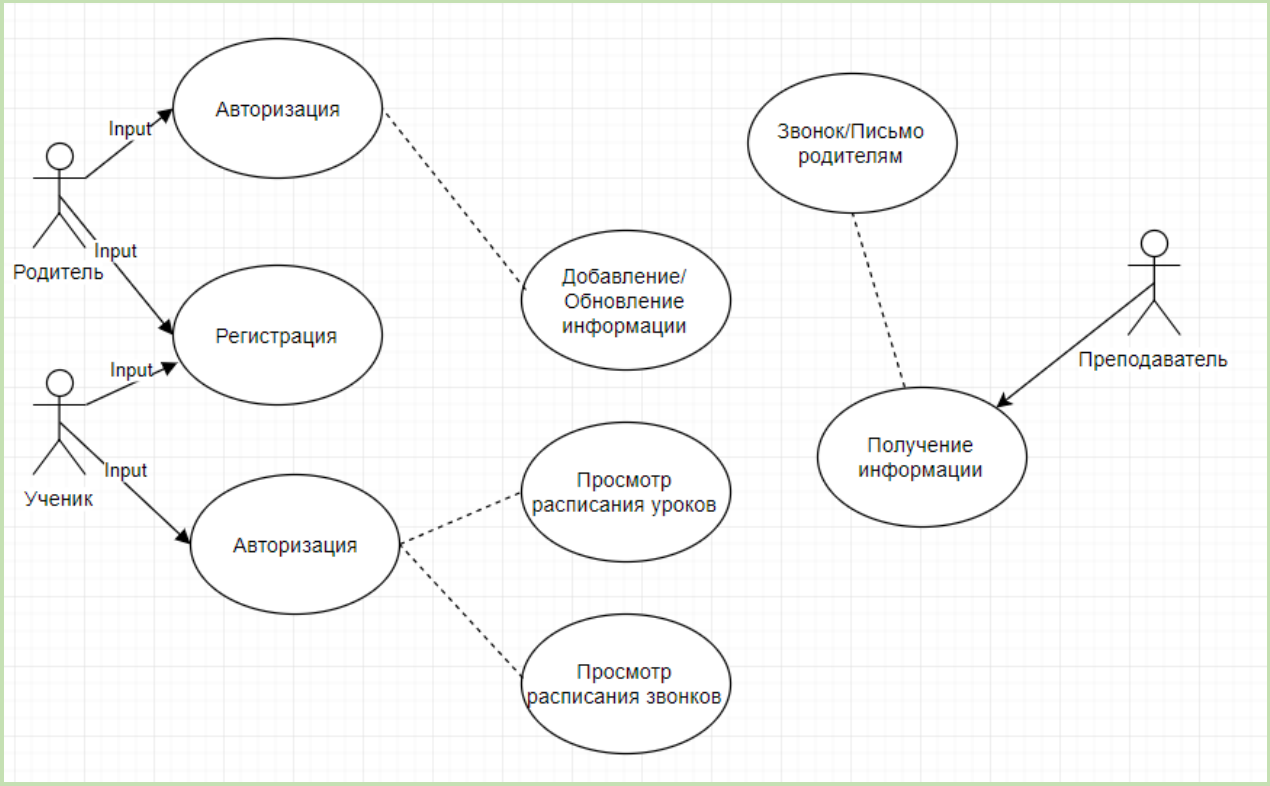


Рисунок 3.6 – Диаграмма вариантов использования

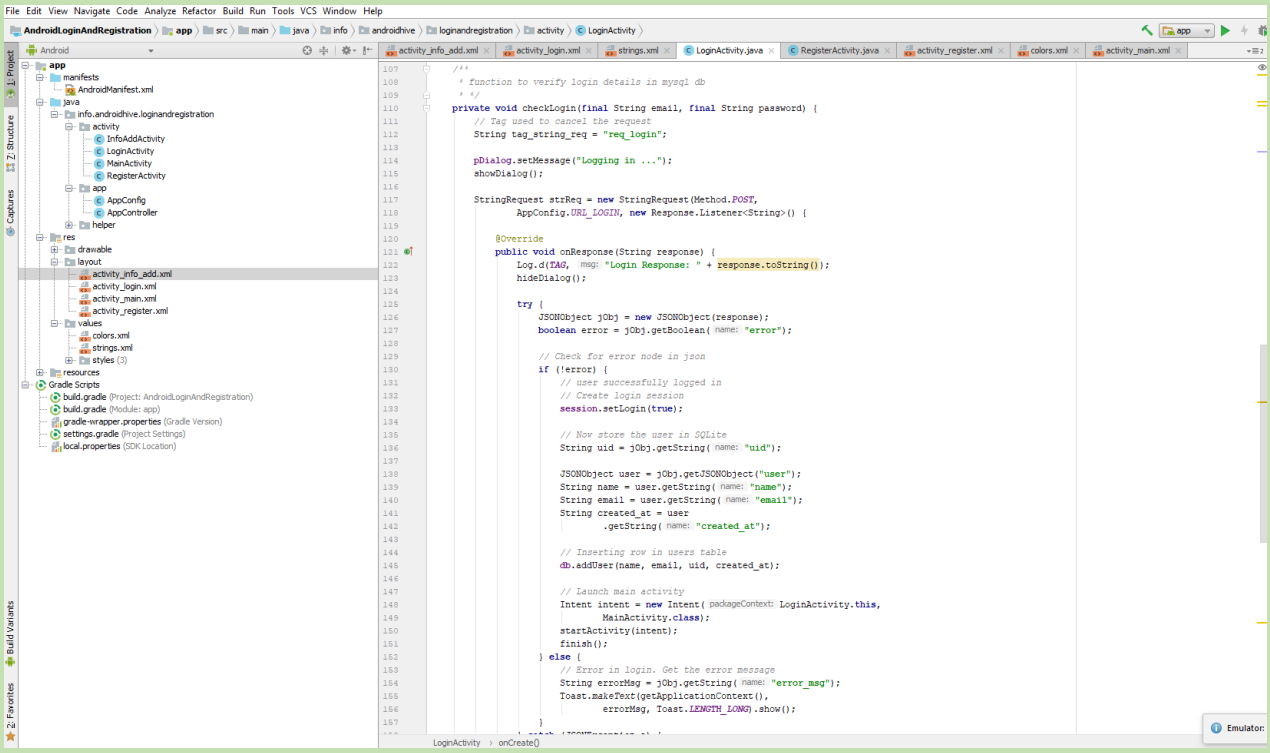


Рисунок 3.7 – Основное окно работы Android studio

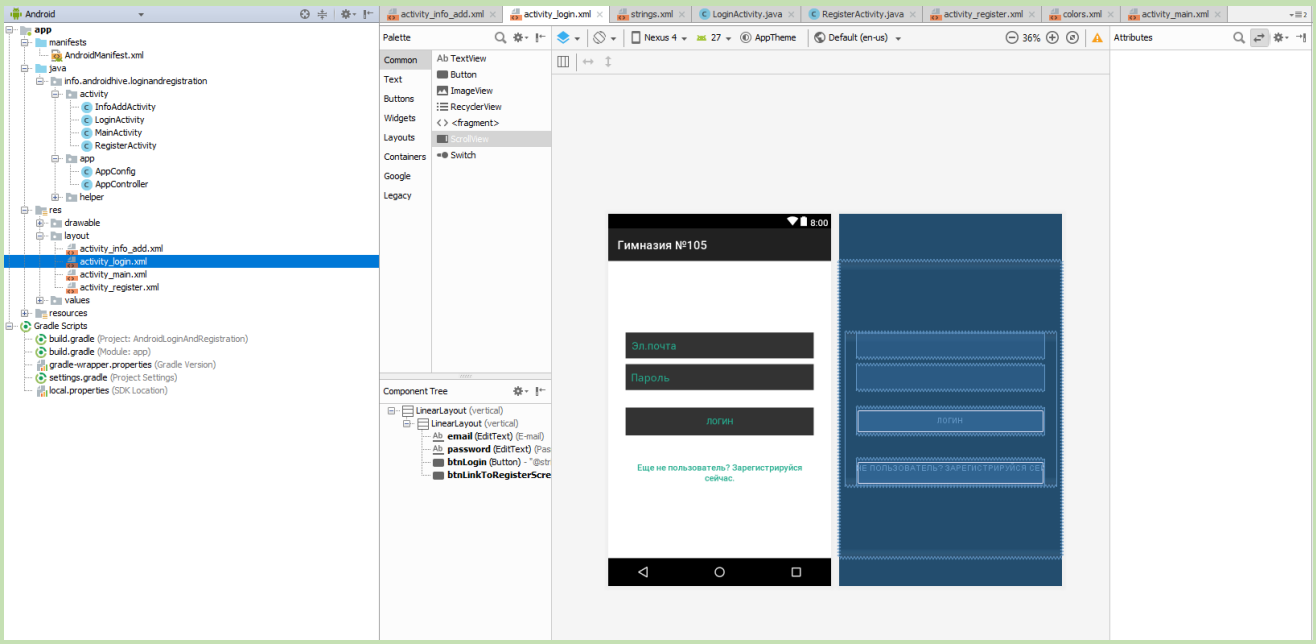


Рисунок 3.8 – Окно работы с графическими

Интерфейс приложения создается при помощи Android studio (рис. 3.7, 3.8). Activity — это компонент приложения, который выдает экран, и с которым пользователи могут взаимодействовать для выполнения каких-либо действий, например, набрать номер телефона, сделать фото, отправить письмо или просмотреть карту. Каждой операции присваивается окно для прорисовки соответствующего пользовательского интерфейса. Обычно окно отображается во весь экран, однако его размер может быть меньше, и оно может размещаться поверх других окон.

Как правило, приложение состоит из нескольких операций, которые слабо связаны друг с другом. Обычно одна из операций в приложении обозначается как «основная», предлагаемая пользователю при первом запуске приложения. В свою очередь, каждая операция может запустить другую операцию для выполнения различных действий. Каждый раз, когда запускается новая операция, предыдущая операция останавливается, однако система сохраняет ее в стеке («стек переходов назад»). При запуске новой операции она помещается в стек переходов назад и отображается для пользователя. Стек переходов назад работает по принципу «последним вошёл — первым вышел», поэтому после того как пользователь завершил текущую операцию и нажал кнопку «Назад», текущая операция удаляется из стека (и уничтожается), и возобновляется предыдущая операция.

Когда операция останавливается по причине запуска новой операции, для уведомления об изменении ее состояния используются методы обратного вызова жизненного цикла операции. Существует несколько таких методов, которые может принимать операция вследствие изменения своего состояния — создание операции, ее остановка, возобновление или уничтожение системой; также каждый обратный вызов представляет возможность выполнить

определенное действие, подходящее для соответствующего изменения состояния. Например, в случае остановки операция должна освободить любые крупные объекты, например, подключение к сети или базе данных. При возобновлении операции вы можете повторно получить необходимые ресурсы и возобновить выполнение прерванных действий. Такие изменения состояния являются частью жизненного цикла операции (см. рис. 3.9).

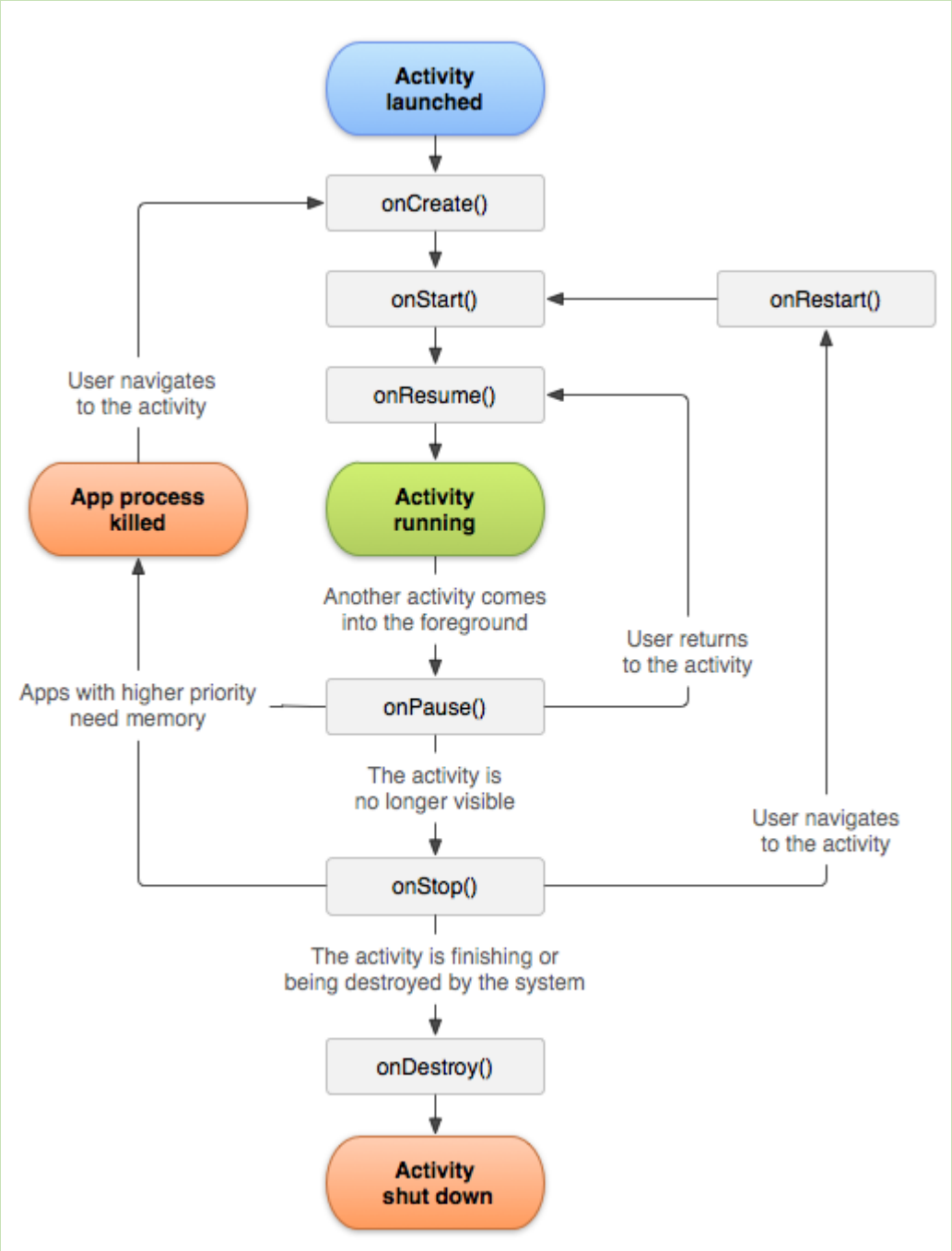


Рисунок 3.9 – Жизненный цикл операции

3.2 Создание и сборка проекта в Android Studio

Для начала создаем новый проект, и в файл build.gradle (рис. 3.10) добавляем поддержку необходимых библиотек путем добавления следующих строк:

```
build.gradle

dependencies {
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    compile 'com.mcxiaoke.volley:library-aar:1.0.0'
}
```

Рисунок 3.10 – Подключение необходимых библиотек

В файлах strings.xml и colors.xml вносятся соответствующие значения(рис. 3.11, 3.12)

```
<resources>

    <string name="app_name">Гимназия №105</string>
    <string name="hint_email">Эл.почта</string>
    <string name="hint_password">Пароль</string>
    <string name="hint_name">ФИО</string>
    <string name="btn_login">Логин</string>
    <string name="btn_register">Регистрация</string>
    <string name="btn_link_to_register">Еще не пользователь? Зарегистрируйся сейчас.</string>
    <string name="btn_link_to_login">Уже зарегистрирован? Авторизуйся.</string>
    <string name="welcome">Добро пожаловать</string>
    <string name="btn_logout">Выйти</string>
    <string name="name">ФИО</string>
    <string name="continueButton">Добавить информацию</string>
    <string name="savebutton">Сохранить</string>

</resources>
```

Рисунок 3.11 – значения, отображаемые на главных экранах в приложении

```xml
<resources>

    <color name="bg_login">#26ae90</color>
    <color name="bg_register">#2e3237</color>
    <color name="bg_main">#428bca</color>
    <color name="white">#ffffff</color>
    <color name="input_login">#222222</color>
    <color name="input_login_hint">#999999</color>
    <color name="input_register">#888888</color>
    <color name="input_register_bg">#3b4148</color>
    <color name="input_register_hint">#5e6266</color>
    <color name="btn_login">#26ae90</color>
    <color name="btn_login_bg">#eceef1</color>
    <color name="lbl_name">#333333</color>
    <color name="btn_logut_bg">#ff6861</color>

</resources>
```

Рисунок 3.12 – значения, которые задают основную палитру цветов приложения

Создаем класс AppConfig.java (рис. 3.14), в котором мы прописываем ссылки на авторизацию и регистрацию. Во время тестирования необходимо указать свой IP адрес, который можно найти с помощью командной строки и команды ipconfig (рис. 3.13):

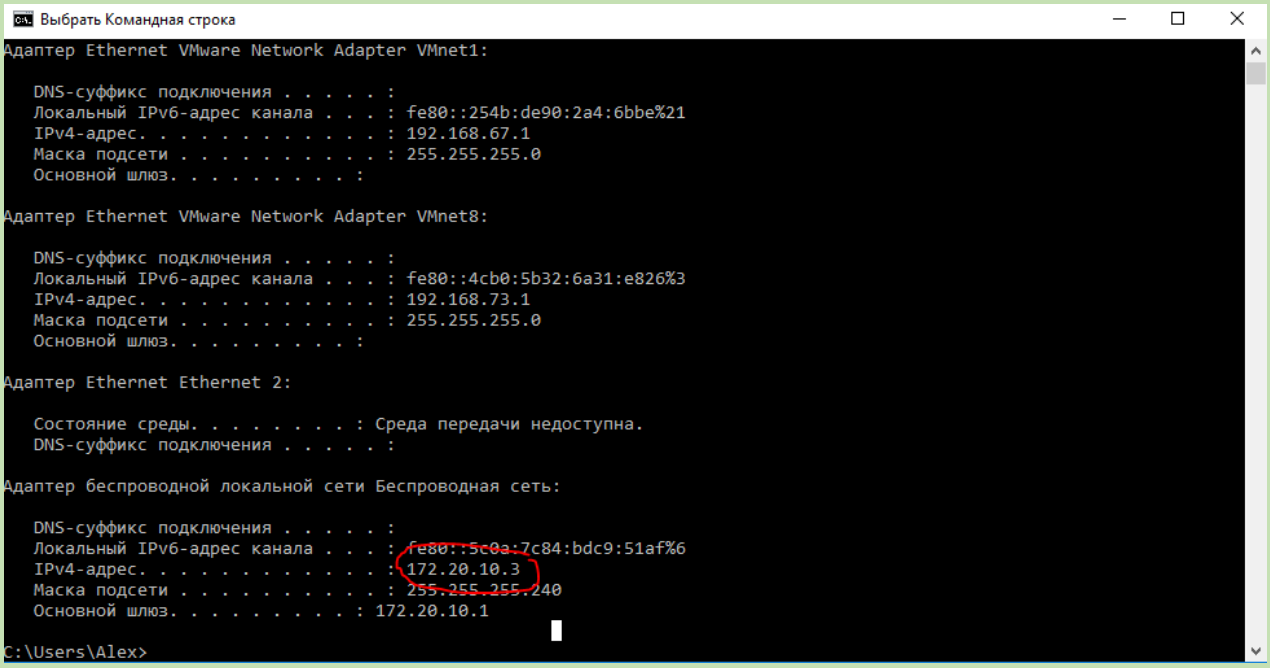


Рисунок 3.13 – Определение IP адреса, для корректной связи с БД

```java
public class AppConfig {
    // Server user login url
    public static String URL_LOGIN = "http://172.20.10.3/android_login_api/login.php";

    // Server user register url
    public static String URL_REGISTER = "http://172.20.10.3/android_login_api/register.php";

    // Server user add info url
    public static String URL_ADDINFO = "http://172.20.10.3/android_login_api/addinfo.php";
}
```

Рисунок 3.14 – Листинг класса AppConfig.java

Класс AppController.java наследуется классом Application, который должен быть запущен при активации приложения. В этом классе мы инициируем все объекты ядра потока.

Далее открываем AndroidManifest..xml (рис. 3.15) и добавляем Internet разрешение. Также добавляется AppController и прописываются активити, которые будут созданы. В данном файле можно выбрать активити по умолчанию, которое будет отображаться при запуске программы.

```xml
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<application
    android:name=".app.AppController"
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="Гимназия №105"
    android:theme="@style/AppTheme">
    <activity
        android:name=".activity.LoginActivity"
        android:label="Гимназия №105"
        android:launchMode="singleTop"
        android:windowSoftInputMode="adjustPan">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".activity.RegisterActivity"
        android:label="@string/app_name"
        android:launchMode="singleTop"
        android:windowSoftInputMode="adjustPan" />
    <activity
        android:name=".activity.MainActivity"
        android:label="Гимназия №105"
        android:launchMode="singleTop" />
    <activity android:name=".activity.InfoAddActivity"></activity>
</application>
```

Рисунок 3.15 – Листинг файла AndroidManifest.xml

Далее добавляем экран авторизации, путем создания самого экрана и добавления кода, чтобы сделать запрос к PHP, MySQL серверу. Создаем файл xml – activity_login.xml в папке: res/layout (рис. 3.16).

```
<EditText
    android:id="@+id/email"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:background="@color/white"
    android:hint="Эл.почта"
    android:inputType="textEmailAddress"
    android:padding="10dp"
    android:singleLine="true"
    android:textColor="@color/input_login"
    android:textColorHint="@color/input_login_hint" />

<EditText
    android:id="@+id/password"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:background="@color/white"
    android:hint="Пароль"
    android:inputType="textPassword"
    android:padding="10dp"
    android:singleLine="true"
    android:textColor="@color/input_login"
    android:textColorHint="@color/input_login_hint" />

<!-- Login Button -->

<Button
    android:id="@+id/btnLogin"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dip"
    android:background="@color/btn_login_bg"
    android:text="Логин"
    android:textColor="@color/btn_login" />

<!-- Link to Login Screen -->

<Button
    android:id="@+id/btnLinkToRegisterScreen"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dip"
    android:background="@null"
    android:text="Еще не пользователь? Зарегистрируйся сейчас."
    android:textAllCaps="false"
    android:textColor="@color/white"
    android:textSize="15dp" />
```

Рисунок 3.16 – Листинг файла activity_login.xml

В приложениях Android визуальный интерфейс нередко загружается из специальных файлов xml, которые хранят разметку. Эти файлы являются ресурсами разметки. Подобный подход напоминает создание веб-сайтов, когда интерфейс определяется в файлах html, а логика приложения - в коде javascript.

Объявление пользовательского интерфейса в файлах XML позволяет отделить интерфейс приложения от кода. Что означает, что мы можем изменять определение интерфейса без изменения кода java. Например, в приложении могут быть определены разметки в файлах XML для различных ориентаций монитора, различных размеров устройств, различных языков и т.д. Кроме того, объявление разметки в XML позволяет легче визуализировать структуру интерфейса и облегчает отладку.

Создаем класс активити LoginActivity.java в папке activity. В этом файле checkLogin() – метод (рис.3.17), который проверяет данные входа в систему на сервере, делая HTTP – запрос.

```
private void checkLogin(final String email, final String password) {
    // Tag used to cancel the request
    String tag_string_req = "req_login";

    pDialog.setMessage("Logging in ...");
    showDialog();

    StringRequest strReq = new StringRequest(Method.POST,
            AppConfig.URL_LOGIN, new Response.Listener<String>() {

        @Override
        public void onResponse(String response) {
            Log.d(TAG, msg: "Login Response: " + response.toString());
            hideDialog();

            try {
                JSONObject jObj = new JSONObject(response);
                boolean error = jObj.getBoolean( name: "error");

                // Check for error node in json
                if (!error) {
                    // user successfully logged in
                    // Create login session
                    session.setLogin(true);
```

Рисунок 3.17 – Фрагмент файла с методом checkLogin();

Далее по вышеуказанному примеру создаем activity_register.xml в папке res/layout. И создаем активити класс RegisterActivity.java (рис.3.18) в папке activity, в котором реализованы методы:

– registerUser() – который хранит пользователя, передавая имя, адрес электронной почты и пароль на php, сервер mysql;

– db.addUser() – который будет вставлять пользователя в базу данных, как только он будет успешно зарегистрирован.

```
private void registerUser(final String name, final String email,
                          final String password) {
    // Tag used to cancel the request
    String tag_string_req = "req_register";

    pDialog.setMessage("Registering ...");
    showDialog();

    StringRequest strReq = new StringRequest(Method.POST,
            AppConfig.URL_REGISTER, new Response.Listener<String>() {

        @Override
        public void onResponse(String response) {
            Log.d(TAG,  msg: "Register Response: " + response.toString());
            hideDialog();
```

Рисунок 3.18 – Фрагмент кода с использованием метода registerUser

3.3 Тестирование программы

На начальном экране нас встречает экран авторизации, изображенный на рисунке 3.19, где можно ввести свои данные, если вы ранее зарегистрировались. Данные берутся из базы данных и сопоставляются с введенными данными. Если же нет совпадений в бд, выдается всплывающее окно, что пользователь не найден.

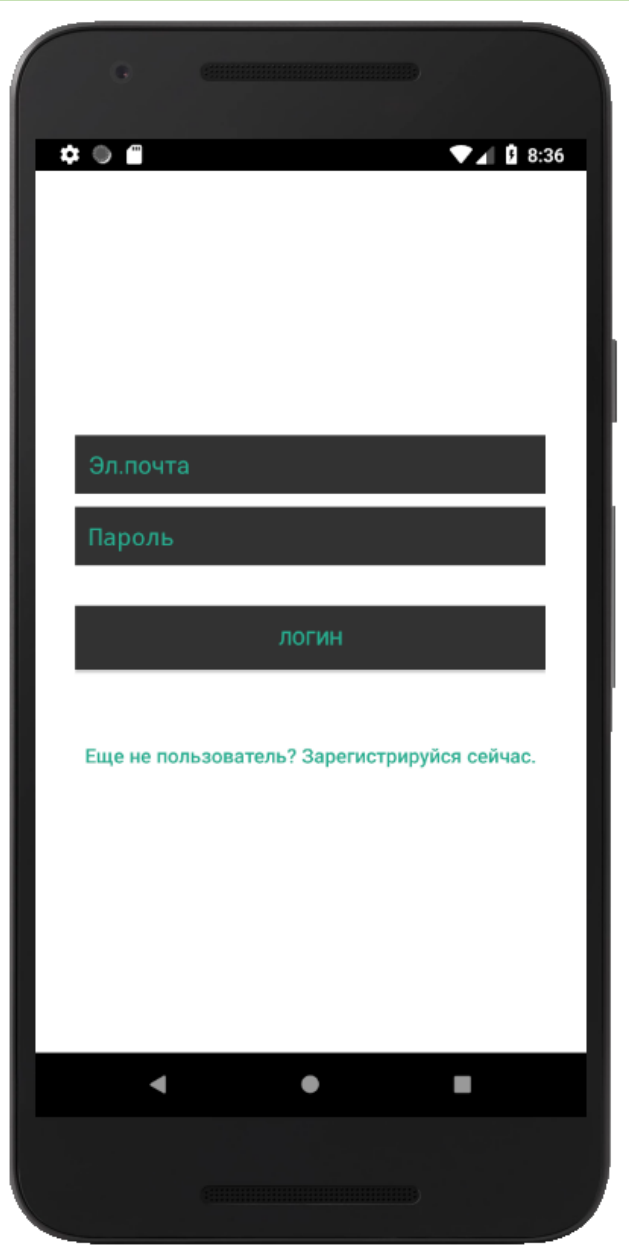


Рисунок 3.19 – Главное окно приложения с авторизацией

В обратном случае при нажатии на надпись: «Зарегистрироваться сейчас» можно пройти в окно регистрации, которое показано на рисунке 3.20.

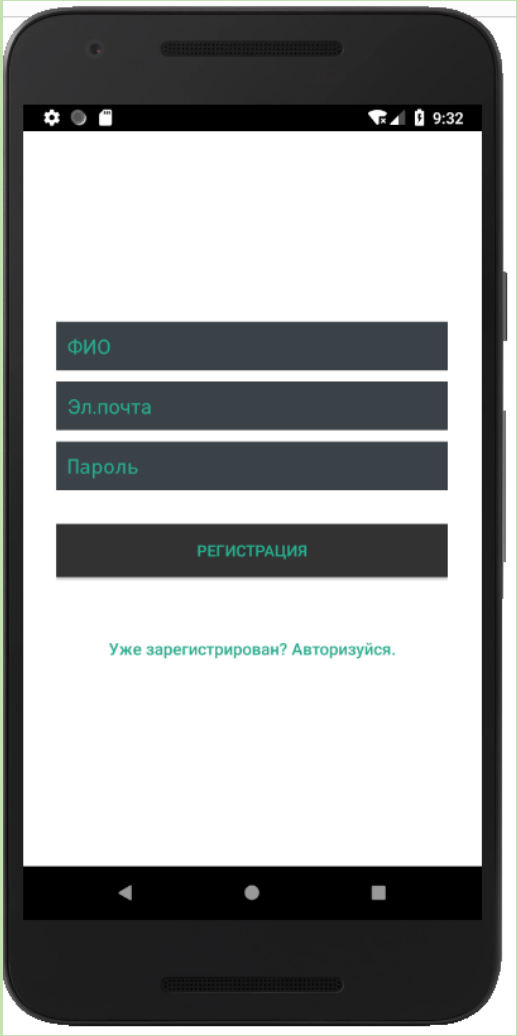


Рисунок 3.20 – Экран регистрации нового пользователя

После прохождения этапа регистрации/авторизации, нас встречает приветственное окно, в котором вас встречает надпись и ваша учетная запись. Можно добавить запрошенную преподавателем информацию, просмотреть расписание звонков и уроков. А также

Далее, на рисунке 3.21 представлены экраны приветствия и добавления информации.

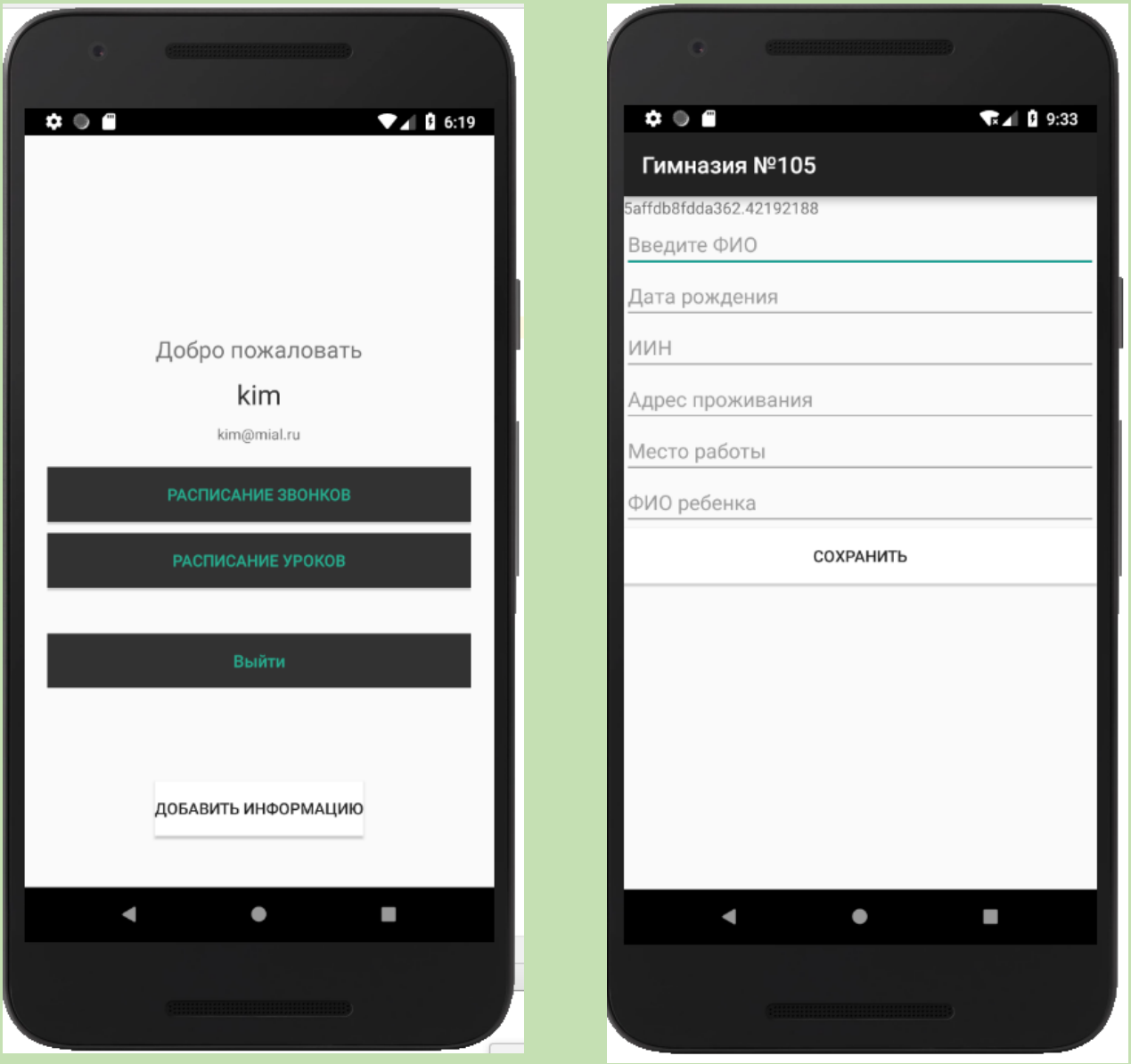


Рисунок 3.21 – Экраны приветствия и добавления информации.

После добавления всей необходимой информации и нажатия кнопки сохранения, появляется всплывающее окно, которое оповещает вас об успешном добавлении/обновлении информации (рис. 3.22).

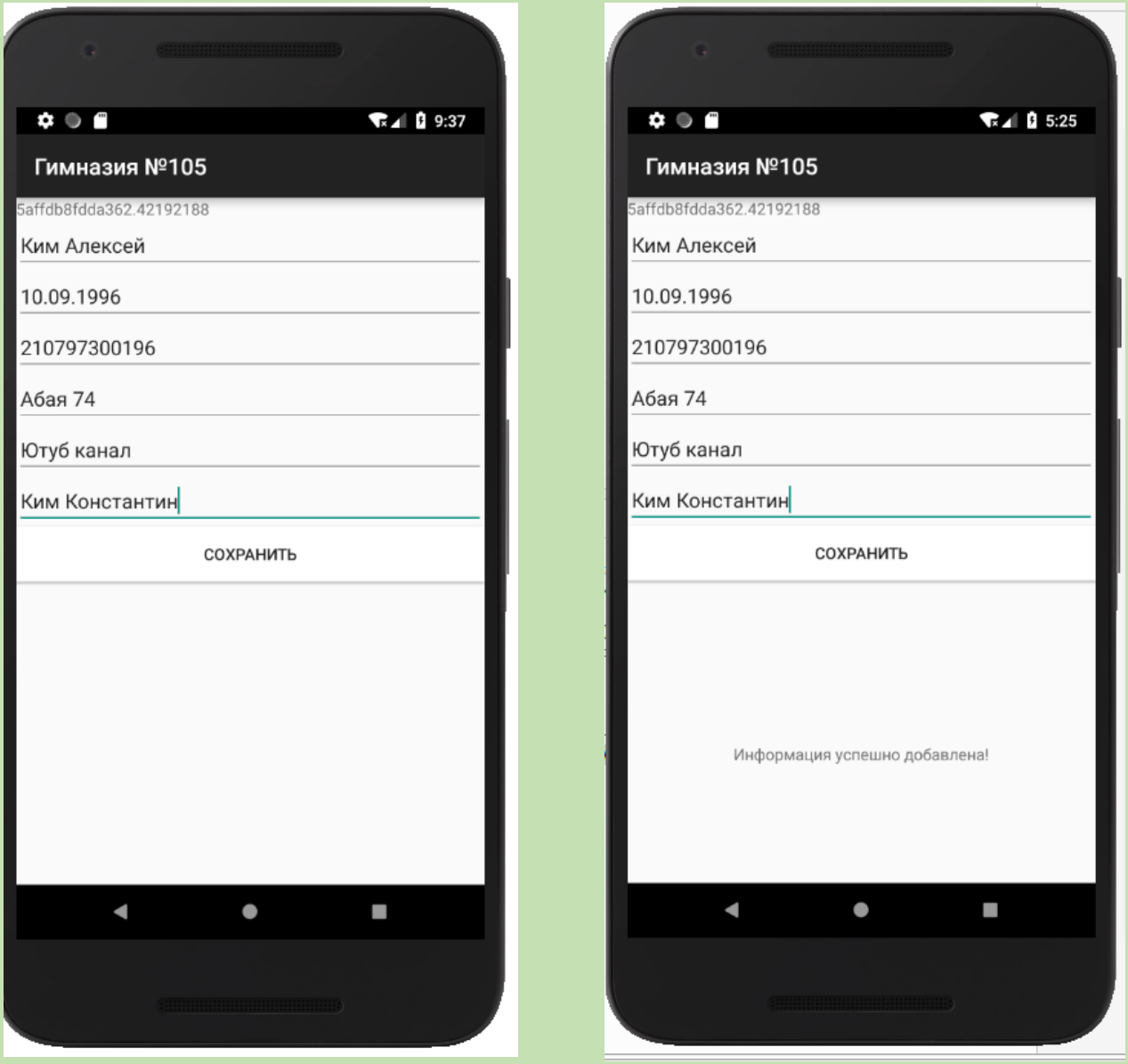


Рисунок 3.22 - Процесс заполнения полей и добавления информации в БД

С экрана приветствия можно перейти к окну расписания звонков (рис.3.23), где можно посмотреть во сколько заканчивается той или иной урок. Данные указаны сразу для обеих смен.

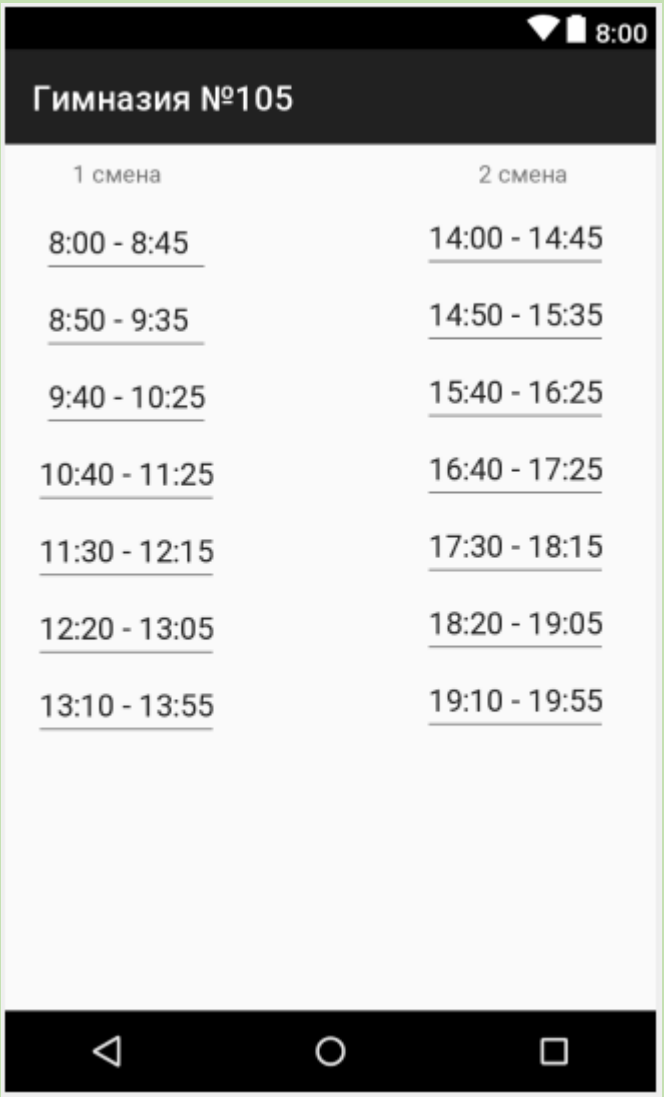


Рисунок 3.23 – Окно просмотра расписания звонков

При переходе в меню просмотра расписания уроков, необходимо выбрать свою параллель и литеру своего класса. Например 9 «Г» (рис. 3.24):

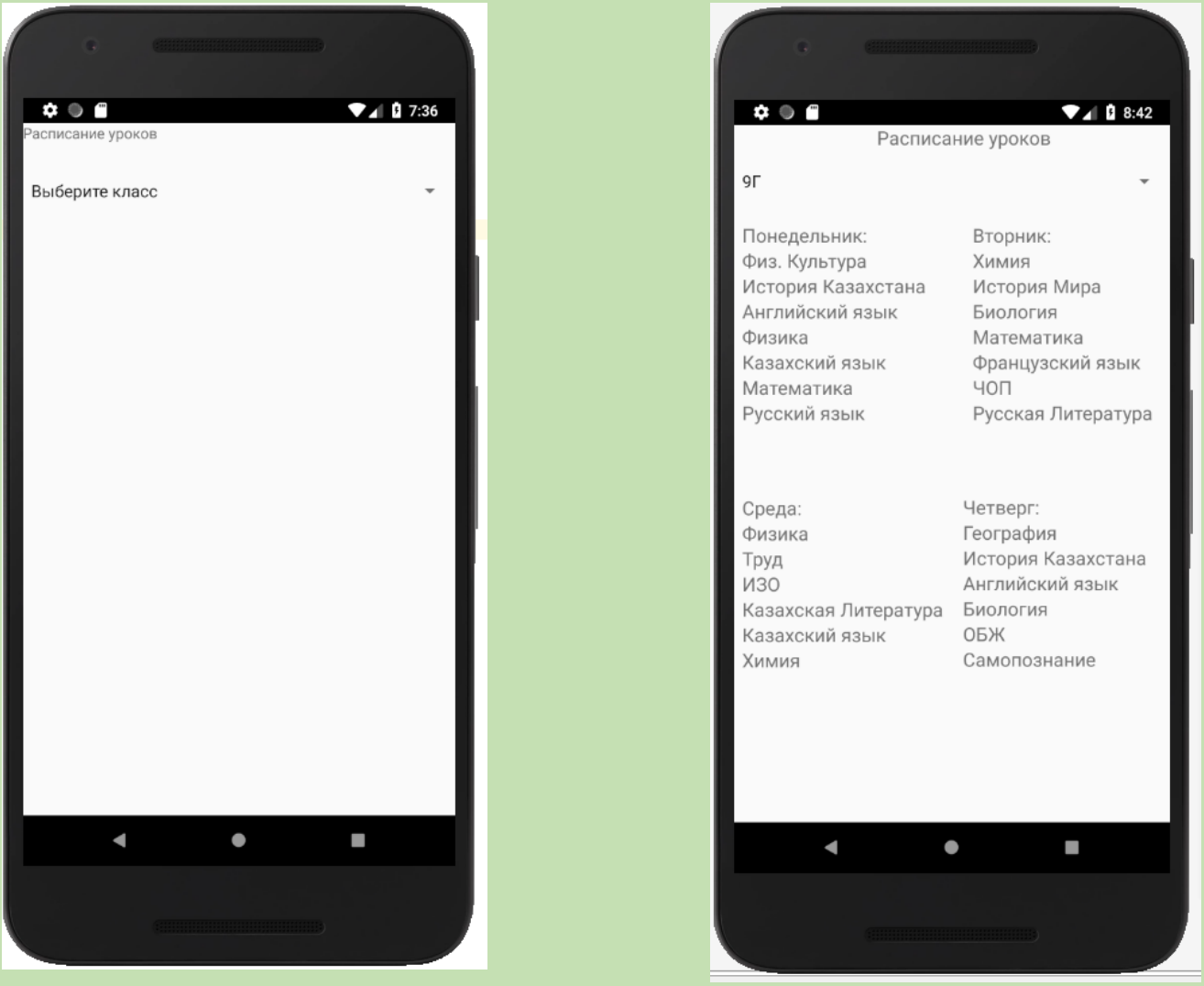


Рисунок 3.24 – Окно выбора класса и просмотра расписания

Как только выбран определенный класс, появляется актуальное расписание на всю неделю.

4 Технико-экономическое обоснование

В данном разделе дипломной работы будут рассмотрены все затраты, необходимые для полноценного функционирования проекта.

Тема данной дипломной работы – «Разработка базы данных школы и приложения для платформы Android».

Данная работа призвана упростить ежегодный процесс сбора личной информации по каждому ученику, повысить скорость реагирования на любые изменения и поддерживать актуальность информации, а также экономия большого количества времени за счет автоматизации и оптимизации процесса сбора.

Цель разработки данного ПО состоит в том, чтобы создать единую базу данных о школьниках и их родителях, а также создание приложения на платформе «Android», для удобного и быстрого взаимодействия с базой, а также достижения наибольшей доступности для всех пользователей, на которых ориентирована разработка.

4.1 Расчет трудоемкости разработки ПО.

Базой для расчета плановой сметы затрат на разработку ПО является объем программного продукта. Общий объем (V_0) программного продукта определяется исходя из количества и объема функций, составляющих данный программный продукт:

$$V_0 = \sum_{i=1}^{n} V_i, \quad (4.1)$$

где Vi − объем отдельной функции ПО;

n − общее число функций.

Из приложения А устанавливаем объем ПО (строки исходного кода, LOC) составляет 14100. Принимаем: $V_0 = 14100$.

Общая трудоемкость процесса рассчитывается по формуле:

$$T_0 = К_с * К_т * К_m * К_н, \quad (4.2)$$

где $К_с$ − коэффициент, учитывающий сложность ПО;

$К_т$ − поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;

$К_н$ − коэффициент, учитывающий степень новизны ПО.

Коэффициент сложности базы данных и приложения определяется на основе данных из таблицы 4.1 и составляет 0,12 так как данной работе присуще две основные характеристики: высокий уровень языкового интерфейса с пользователем и режим работы в реальном времени.

Таблица 4.1 – Дополнительные коэффициенты сложности ПО

Характеристика ПО	Значения $К_с$
1. Функционирование ПО в расширенной операционной среде (связь с другими ПО)	0,08
2. Интерактивный доступ	0,06
3. Обеспечение хранения, ведения и поиска данных в сложных структурах	0,07
4. Наличие у ПО одновременно нескольких характеристик по табл.Г4.1, приложение Г	
4.1 2 характеристики	0,12
4.2 3 характеристики	0,18
4.3 Свыше 3-х характеристик	0,26

Поправочный коэффициент, учитывающий степень использования при разработке ПО основных модулей ($К_т$), определяется исходя из данных таблицы 4.2 и составляет 0,7.

Таблица 4.2 – Значения поправочного коэффициента, учитывающего использование стандартных модулей типовых программ и ПО ($К_т$)

Степень охвата реализуемых функций разрабатываемого ПО стандартными модулями, типовыми программами и ПО	Значения $К_т$
1. От 60 % и выше	0,6
2. От 40 % до 60	0,7
3. От 20 % до 40 %	0,8
4. До 20 %	0,9
5. Типовые программы и ПО не используемые для реализации функций разрабатываемого ПО	1,0

Поправочный коэффициент, учитывающий новизну разрабатываемого ПО ($К_н$) определяется на основе данных таблицы 4.3 и соответственно составляет 1,0.

Таблица 4.3 – Поправочные коэффициенты, учитывающие новизну ПО ($К_н$)

Категория новизны	Степень новизны	Использование		Значение $К_н$
		На основе нового типа ПК	В среде новой ОС	
А	Принципиально новые ПО, не имеющие доступных аналогов	+ – + –	+ + – –	1,75 1,6 1,2 1,0
Б	ПО, являющиеся развитием определенного параметрического ряда ПО	+ – +	+ – –	1,0 0,9 0,8
В	ПО, являющиеся развитием определенного параметрического ряда ПО, разработанных для ранее освоенных типов конфигурации ПК и ОС	–	–	0,7

Базисом для определения нормативной трудоемкости при разработке указанного проекта являются, заявленные нормы времени на разработку проекта в зависимости от точечного объема программного обеспечения и соответствующей ему группе сложности.

Нормативная трудоемкость проекта ($Т_н$) в свою очередь определяется на основе принятого в расчет объема ПП и категории сложности, которая уточняется с учетом сложности и новизны проекта и степени использования стандартных модулей при разработке.

Учитывая данные из Приложения А: для 1–ой категории сложности ПО $Т_н$ =456.

Рассчитаем общий объем трудоемкости по формуле (4.2):

$$Т_0 = 456 * 0{,}12 * 0{,}7 * 1 = 38{,}30.$$

На основе трудоемкости определяются плановое число разработчиков ($Ч_р$) и плановые сроки, необходимые для проекта($Т_р$). При этом могут решаться нижеперечисленные задачи:

– расчет числа исполнителей при заданных сроках разработки проекта;
– определение сроков разработки проекта при заданной численности
– исполнителей.

Численность исполнителей проекта ($Ч_р$) рассчитывается по следующей формуле:

$$Ч_р = Т_0/(Т_р*Ф_{эф}), \quad (4.3)$$

где $Ф_{эф}$ − эффективный фонд времени работы одного работника в течении года (дн.);

$К_0$ − общая трудоемкость разработки проекта (чел./дн.);

$Т_р$ − срок разработки проекта (лет).

Срок разработки базы данных и приложения ($Т_р$) определяется по формуле:

$$Т_р = Т_0/(Ч_р*Ф_{эф}), \quad (4.4)$$

где $Ч_р$ – плановое число разработчиков.

Эффективное время работы одного работника ($Ф_{эф}$) рассчитывается по данной формуле:

$$Ф_{эф} = Д_г+Д_п+Д_в+Д_о, \quad (4.5)$$

где $Д_г$ − количество дней в году;

$Д_п$ − количество праздничных дней в году;

$Д_в$ − количество выходных дней в году;

$Д_о$ − количество дней отпуска.

Сопоставив данные с производственным календарем на 2018 год.: $Д_г$ − 365; $Д_п$ = 16; $Д_в$ – 103; $Д_о$ – 10, эффективный фонд времени одного кадра должно составить:

$$Ф_{эф}=365−16−103−10 = 236 \text{ дня}$$

Плановое число разработчиков программного обеспечения $Ч_р$ = 1,следовательно, по формуле (4.4)

$$Т_р=38,30/1*236=0.16 \text{ лет}$$

Таким образом, учитывая произведенные расчеты и формулу (4.3) имеем следующие данные:

$$Ч_р=38{,}30/0.16*236=1 \text{ чел}$$

4.2 Расчет затрат на разработку информационных технологий

Под информационными технологиями понимаются экономические информационные системы (ЭИС), программные продукты (ПП), информационные базы данных и т.д.

Расчет полных затрат на разработку проектного решения в виде информационных технологий ($С_{пі}$) осуществляется по формуле:

$$С_{пі} = З_{фот} + З_{сзі} + М_і + Р_{сі} + Р_{мі} + Р_{нкі} + П_{зі} + Р_{ні}, \quad (4.6)$$

где $З_{фот}$ – общий фонд оплаты труда разработчиков, тенге;

$З_{сзі}$ – отчисления по социальному налогу, тенге;

$М_і$ – затраты на материалы, тенге;

$Р_{сі}$ – затраты на специальные программные средства, необходимые для разработки проектного решения, тенге;

$Р_{мі}$ – затраты, связанные с эксплуатацией техники, тенге;

$П_{зі}$ – прочие затраты, тенге;

$Р_{ні}$ – накладные расходы, тенге.

Размер фонда оплаты труда разработчиков ($З_{ФОТ}$) рассчитывается по формуле:

$$З_{ФОТ} = З_о + З_д, \quad (4.7)$$

где $З_о$ – основная заработная плата, тенге;

$З_{д.}$ – дополнительная заработная плата, тенге.

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле:

$$З_о = \sum_{i=1}^{n} Т_{чi} * Т_ч * Ф_n * К, \quad (4.8)$$

где n − количество исполнителей, занятых разработкой конкретного ПО;

$Т_{чі}$ − часовая тарифная ставка і–го исполнителя (тыс. тенге);

$Ф_п$ − плановый фонд рабочего времени і–го исполнителя (дней), 21 раб день в месяц;

$Т_ч$ − количество часов работы в день (час), 8 часов;

К − коэффициент премирования, составляет 1,5.

По данным о специфике и сложности выполняемых функций составляется штатное расписание группы специалистов-исполнителей, участвующих в разработке ПО, с определением образования, специальности, квалификации и должности (таблица 4.4).

Таблица 4.4 – Сведения по работникам, задействованным в проекте

Специалист - Исполнитель	Количество, человек	Заработная плата в месяц, тенге
IT специалист	1	150000
Итого	1	150000

Часовая тарифная ставка рассчитывается путем деления месячнойтарифной ставки, установленную при 40–часовой недельной норме рабочего времени и общего фонда времени($Ф_р$)

$$Т_ч=Т_м/Ф_р, \quad (4.9)$$

где $Т_ч$ – часовая тарифная ставка (тыс. тенге);
$Т_м$ – месячная тарифная ставка (тыс. тенге).

Общий фонд времени:

$$Ф_р=Т_ч*Ф_п, \quad (4.10)$$

Таким образом:

$$Ф_р=8*21=168 \text{ ч.}$$

Рассчитаем тарифную ставку инженера-программиста:

$$Т_ч =150\ 000/168=893 \text{ тг. в час}$$

Основываясь формулой (4.8) основная заработная плата инженера-программиста соответственно следующая:

$$З_о=893*8*21*1.5=225\ 036 \text{ тг.}$$

Дополнительная заработная плата составляет 10% от основной заработной платы и рассчитывается по следующей формуле:

$$З_д=З_о*Н_д/100, \quad (4.11)$$

где $Н_д$ – коэффициент дополнительной заработной платы разработчиков 23%.

$$З_д=225\ 036*0,23=51\ 758тг$$

Социальный налог составляет 11% (ст. 358 п. 1 НК РК) от доходаработника, и рассчитывается по формуле:

$$З_с=(ФОТ-П)*11\%, \quad (4.12)$$

где П – пенсионные отчисления, которые составляют 10% от ФОТ и социальным налогом не облагаются

$$П=ФОТ*10\%, \quad (4.13)$$

Таким образом:

$$П=225\ 036*0,1=22\ 504\ тг.$$

$$З_с=(225\ 036-22\ 504)*0,11=22\ 279\ тг.$$

Затраты на материалы определяются по формуле:

$$М_i=(З_о*Нмз)100\%, \quad (4.14)$$

где $Н_{мз}$ − норма расхода материалов от основной заработной платы (3–5%).

$$М_i=225\ 036*0,04=9001\ тг.$$

Расходы по статье «Спецоборудование» ($Р_с$) составляют 138 393 тенге без НДС, с учетом НДС $Р_с$ = 155000 тенге. Сводные затраты по стоимостиоборудования представлены в таблице 4.5

Таблица 4.5 – перечень оборудования, используемого при реализации проекта

Оборудование	Марка, характеристики	Кол-во единиц	Стоимость, тг	Общая сумма, тг
Ноутбук	Lenovo Y500, Intel Core i5 3230M, 2,60Ghz, RAM 8gb, ROM 1tb.	1	155000	155000
Итого				155000

Расходы по части «Машинное время» $Р_м$ включают оплату машинноговремени, который необходимо для реализации и отладки программы, определяющийся по нормативам (в машино – часах) на 100 строк исходного кода ($Н_{мв}$) машинного времени в зависимости от характера решаемых задач и типа персонального компьютера:

$$Р_м = Ц_м \cdot (V_о / 100) \cdot Н_{мв}, \quad (4.15)$$

где $Ц_м$ − цена одного машинного часа (тыс.тенге);

$Н_{мв}$ − норматив расхода машинного времени на отладку 100 строкисходного кода (машинно–часов);

$V_о$ − общий объем ПО (строк исходного кода).

Норматив расхода машинного времени на отладку 100 строк исходника определяется на основе таблицы Д.1 (Приложения Д) и составляет 12 ч/100 строк кода

$$Р_м = 893 \cdot (14100/ 100) \cdot 12 = 1\ 510\ 956 \text{ (тенге)}$$

Расходы по статье «Прочие затраты» ($П_з$) на конкретное продукт, либосервис включают затраты на приобретение и подготовку специальной научно–технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по организации, в процентах к основнойзаработной плате:

$$П_з = З_о * Н_{пз} / 100, \quad (4.16)$$

где $Н_{пз}$ − норматив прочих затрат в целом по организации (20%)

Таким образом:

$$П_з = 225\ 036 * 0{,}2 = 45\ 007 \text{тг.}$$

Затраты по статье «Накладные расходы» ($Р_н$), рассчитывается по нормативу ($Н_{рн}$) в процентном отношении к основной заработной платеисполнителей. Норматив устанавливается в целом по организации:

$$Р_н = З_о * Н_{рн} * 100\%, \quad (4.17)$$

где $Р_н$ – накладные расходы на конкретную ПО (тыс.тенге);

$Н_{рн}$ – норматив накладных расходов в целом по организации (70%).

Таким образом накладные расходы составят:

$Р_н$=225 036∗0,7=157 525тг.

В соответствие с формулой (4.6) полные затраты на разработкуинформационной системы «Экономического отдела», составит:

Сп*i*=225 036+22 279+9 001+155 000+ 1 510 956+45 007+157 525=2 124 804тг

Таблица 4.6 – Затраты на разработку

Затраты на разработку	Условное обозначение	Значение, тенге	В процентах от общей суммы
Фонд оплаты труда	$З_{ФОТ}$	225 036	11
Социальный налог	$З_{сзі}$	22 279	1
Материалы	$М_і$	9 001	0
Спецоборудование	$Р_{сі}$	155 000	7
Машинное время	Рмі	1 510 956	71
Прочие затраты	$П_{зі}$	45 007	2
Накладные расходы	$Р_{ні}$	157 525	7
Итого:		2 124 804	100

Рисунок 4.1 – Структура затрат на разработку программного продукта

4.3 Расчет цены программного продукта

Расчет цены ПП, который разработан одной организаций по заказу другой и не предназначен для тиражирования, осуществляется по формуле

$$Ц_{пп} = З_{рпр} + П_{п} + НДС, \quad (4.18)$$

где $Ц_{пп}$ – цена программного продукта, тенге;

$З_{рпр}$ – затраты на разработку проектного решения, в данном случае программного продукта, тенге;

$П_{п}$ – планируемая прибыль, тенге;

НДС – налог на добавленную стоимость, тенге.

Планируемая прибыль составляет (20%) от себестоимости разработки.

$$П_{п} = 2\ 124\ 804 * 0{,}2 = 424\ 961 \text{ тг.}$$

НДС, начисленный на ПП, определяется следующим образом

$$НДС = (З_{рпр} + П_{п}) * k НДС, \quad (4.19)$$

где kНДС – ставка налога на добавленную стоимость.

Подставив данные в формуле (4.19) получаем:

$$НДС = (2\ 124\ 804 + 424\ 961) * 0{,}12 = 305\ 972 \text{ тг.}$$

Подставив данные в формуле (4.18) получаем:

$$Ц_{пп} = 2\ 124\ 804 + 424\ 961 + 305\ 972 = 2\ 855\ 737 \text{ тг}$$

4.4 Вывод по технико – экономической части

Создание базы данных и мобильного приложения для платформы Android, является безусловно инновационным и нужным проектом. В технико – экономической части данной дипломной работы произведен расчет всех издержек и расходов на осуществления данного проекта. Себестоимость работы составила 2 124 804 тенге. Цена реализации программного продукта с учетом НДС составляет 2 855 737тг. Эффективность данного проекта можно будет определить более точно после внедрения ПП.

С внедрением данного ПП мы сделаем еще один шаг на встречу инновационным технологиям, в десятки раз уменьшится время на сбор основной информации о школьниках и их родителях, часть нагрузки с преподавателей также будет снята, что позволит повысить продуктивность и улучшить качество их работы. Хранение информации в единой базе позволит

избежать ненужной бумажной волокиты и сэкономит огромное количество времени при поиске и обновлении информации.

5 Безопасность жизнедеятельности

В рамках данного дипломного проекта рассмотрены вопросы автоматизации процесса сбора информации и поддержания ее актуальности. С внедрением данного программного продукта возрастет продуктивность работы персонала, отвечавших за сбор информации, и позволит повысить их продуктивность в плане обучения.

Цель проекта – сократить время сбора различной информации о родителях, что позволит поддерживать актуальность информации на высоком уровне, оперативно реагировать на изменения личной информации и экономить рабочие ресурсы учителей.

5.1 Анализ условий труда в здании КГУ гимназии №105

Описание главного здания школы (рис.5.1):

– основное здание трехэтажное;

– здание поделено на левое и правое крыло, которые соединены коридором;

– лестницы расположены в левых и правых частях коридора;

– количество кабинетов на этаже, учитывая и лево и правое крыло – 12;

– имеется 1 главный вход/выход и 3 запасных входа/выхода;

Таблица 5.0 – размеры участков, принятых по плану

Номер участка	Наименование	Длина(м)	Ширина(м)
1	Класс	8	6
2	Коридор	8	3
3	Коридор	8	6
4	Расстояние до дверного проема (Дверной проем принимаем за 0)	3	2
5	Лестница с 3 на 2 этаж	12	2
6	Лестница со 2 на 1 этаж	12	2
7	Лестница к выходу	2	2

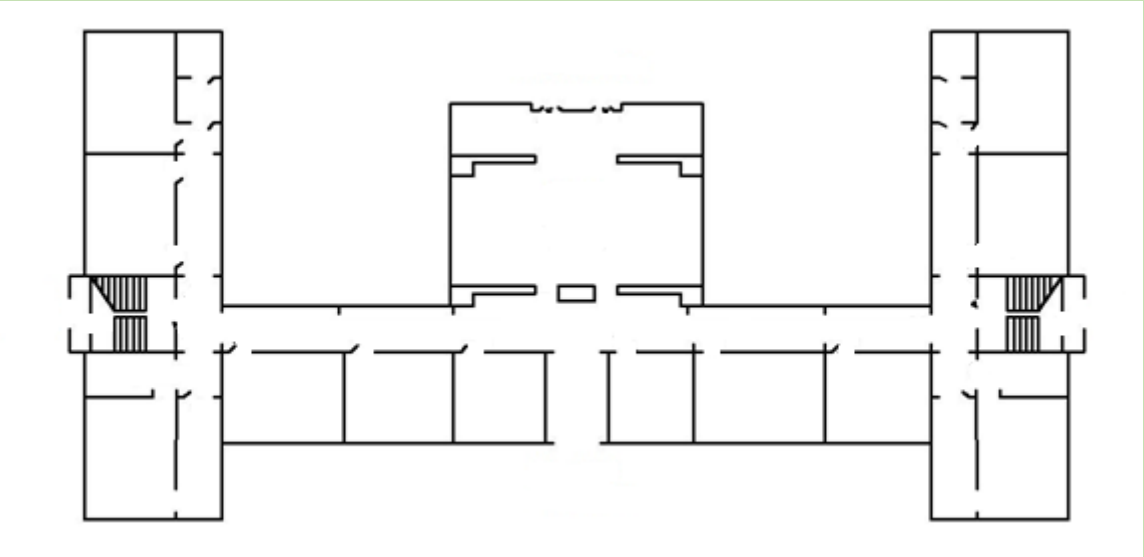


Рисунок 5.1 – Общая схема этажности основного здания КГУ Гимназии №105

Причины возникновения пожаров:

– несоблюдение правил эксплуатации учебного оборудования и электрических устройств;
– неосторожное обращение с огнём;
– самовозгорание веществ и материалов;
– поджоги;
– неправильное пользование газовым оборудованием;
– солнечный луч, действующий через различные оптические системы;

При обнаружении очага возгорания, можно попытаться самостоятельно потушить его с помощью огнетушителей или первичных средств, расположенных на пожарных стендах, щитах либо же подручными средствами. В случае неудачной попытки незамедлительно принять меры по информированию персонала и пожарных служб, а затем приступить к краткому инструктированию и немедленной эвакуации. Важность инструктирования нельзя недооценивать: ни в коем случае нельзя поддаваться панике, правильное поведение преподавателя и грамотные действия со стороны персонала помогут избежать неадекватного поведения среди учеников.

В случае землетрясения следует помнить, что первые толчки самые сильные (длительность от 5 секунд до минуты). Если покинуть помещение в безопасности не представляется возможным, следует занять наиболее безопасные места: под партой, под столом, в дверном проеме и т.д. И дождавшись подходящего момента в организованном порядке покинуть помещение согласно плану эвакуации.

В соответствии с предопределенными правилами и положениями предусмотрены маршруты эвакуации, которые позволяют эвакуировать в требуемое время и эвакуировать всех лиц в зданиях. Время, необходимое для

того, чтобы люди покидали здание или здание, называется поселением. Период для безопасной эвакуации по-прежнему возможен, что необходимо.

Для безопасной эвакуации людей из зданий и помещений ожидаемое время эвакуации должно быть меньше времени, необходимого для эвакуации. Необходимое время эвакуации является табличным значением, тогда как расчетное время рассчитывается из расчета времени для перемещения одного или нескольких потоков человека из наиболее депонированных человеческих объектов. Тип движения во время эвакуации делится на секции (коридор, вход, лестница, пассажир и т.д.).

5.2 Расчет эвакуационных путей.

При определении расчетного времени длина и ширина каждого участка пути эвакуации принимаются по проекту. Длина пути по лестничным маршам измеряется по длине марша. Длина пути в дверном проеме принимаем за ноль. Расчетное время эвакуации людей (t_p) следует определять, как сумму времени движения людского потока по отдельным участкам пути t_i по формуле:

$$t_p = t_1 + t_2 + t_3 + , \dots , + t_i, \quad (5.1)$$

где t_1 – время движения людского потока на начальном участке, мин;

t_2, t_3,…, t_i – время движения людского потока на последующих участках, мин.

Плотность потока на первом участке пути D1 определяют по формуле:

$$D1 = N1 \cdot f / (l1 \cdot w1), \quad (5.2)$$

где N1 – число людей на первом участке;

f – средняя площадь горизонтальной проекции человека: (взрослого в летней одежде – 0,1, взрослого в зимней одежде – 0,125, подростка – 0,07м^2);

l1 – длина первого участка;

w1 – ширина первого участка;

Интенсивность движения людского потока по каждому из последующих участков, идущих после первого, определяют по формуле:

$$q_i = q_{i-1} \cdot w_{i-1} / w_1, \quad (5.3)$$

где w_i, w_{i-1} – ширина рассматриваемого i-го и предшествующего ему i-1 участка пути, м;

q_i, q_{i-1} – значения интенсивности движения потока по рассматриваемому i и предшествующему i-1 участкам пути, м/мин.

Задание: Определить расчетное время эвакуации людей из коммунального государственного учреждения гимназии № 105. Основное здание гимназии II степени огнестойкости, имеет 3 этажа, 36 кабинетов, каждый из которых рассчитан на 30 ученических мест. Составить полную схему движения при эвакуации.

Наиболее отдаленной точкой от запасных и главного входов является 3 этаж, и так как выходы расположены в центре первого этажа и по левому и правому краю, самый длинный путь эвакуации будет лежать из кабинетов занимающий середину этажа. Основное здание школы является полностью симметричным и при эвакуации кабинеты находящиеся в левом и правом крыле выводятся организованным строем, каждый к своему выходу, расположенному соответственно.

Общее число учеников:

$$N_0 = 36 \cdot 25 = 900 \text{ чел.}$$

Весь эвакуационный путь разобьем на отдельные участки пути. Начальным участком является класс. При определении границ последующих участков на пути движения к эвакуационному выходу исходим из того, что в пределах расчетного участка не должна изменяться ширина пути и не должно быть слияния потоков. При таких условиях принимаем интенсивность и скорость движения постоянными по длине участка.

$$D1 = 30 \cdot 0{,}07 / (8 \cdot 5{,}5) = 0{,}05 \text{ м}^2/\text{ м}^2$$

Значение скорости движения потока людей в зависимости от плотности D, зависимости интенсивного движения людского потока q от его плотности и скорости движения приведены в таблице 4.1.

Таблица 5.1 – Значения скорости и интенсивности движения людского потока по горизонтальному пути в зависимости от плотности.

Плотность потока D, м²/м²	Горизонтальный путь		Дверной проем	Лестница вниз		Лестница вверх	
	скорость v, м/мин	интенсивность q, м/мин	интенсивность q, м/мин	скорость v, м/мин	интенсивность q, м/мин	скорость v, м/мин	интенсивность q, м/мин
0,01	100	1	1	100	1	60	0,6
0,05	100	5	5	100	5	60	3
0,1	80	8	8,7	95	9,5	53	5,3
0,2	60	12	13,4	68	13,6	40	8
0,3	47	14,1	16,5	52	15,6	32	9,6
0,4	40	16	18,4	40	16	26	10,4
0,5	33	16,5	19,6	31	15,5	22	11
0,6	27	16,2	19	24	14,4	18	10,8
0,7	23	16,1	18,5	18	12,6	15	10,5
0,8	19	15,2	17,3	13	10,4	13	10,4
0,9 и более	15	13,5	8,5	8	7,2	11	9,9

Примечание Табличное значение интенсивности движения в дверном проеме при плотности потока 0,9 и более, равное 8,5 м/мин, установлено для дверного проема шириной 1,6 м и более, а при дверном проеме меньшей ширины δ интенсивность движения следует определять по формуле $q = 2{,}5 + 3{,}75\,\delta$

Исходя из данных таблицы 4.1 получаем:

$$v_1 = 100 \text{ м/мин}; \; q_1 = 5\text{м/мин};$$

Интенсивность движения людского потока:

$$q = D \cdot v, \tag{5.4}$$

Таким образом интенсивность движения на первом участке:

$$q_1 = 0{,}05 \cdot 100 = 5 \text{ м/мин}$$

Так как q_i меньше или равно q_{max}, то время движения на участке пути:

$$t_1 = l_1 / v_1, \tag{5.5}$$

$$t_1 = 8 / 100 = 0{,}08 \text{ мин}$$

На втором участке происходит сужение пути, т.е. $w_2 = 3$м;

Интенсивность на втором участке рассчитывается по формуле (5.3) и равна:

$$q_2 = \frac{q_1 * w_1}{w_2} = \frac{4 * 6}{3} = 8\text{м/мин}$$

Значение скорости людского потока берем из таблицы 5.1:

v_2=80м/мин;

И по формуле (5.5) получаем время движения на втором участке пути равное:

$t_2 = 8 / 80 = 0{,}1$ мин

На участке под номером 3 происходит слияние;

При слиянии вначале участка i двух и более людских потоков интенсивность движения (q_i) вычисляется по формуле:

$$q_i = \frac{\sum q_{i-1} * w_{i-1}}{w_1} \quad (5.6)$$

где q_{i-1} – интенсивность движения людских потоков, сливающихся в начале участка i, м/мин.

w_{i-1} – ширина участков пути слияния, м;

w_i – ширина рассматриваемого участка пути, м.

При этом, если значение q_i, полученное из вычислений по формуле (5.6) превышает максимальное значение q_{max}, то ширину w_1 данного участка пути необходимо увеличить так, чтобы соблюдалось условие $q_i \leq q_{max}$.

При невозможности выполнения вышеуказанного условия, интенсивность и скорость движения людского потока по участку пути i определяют при значении D=0,9 и более.

Таким образом получаем интенсивность пути на третьем участке, используя формулу (5.6):

$$q_3 = \frac{q_2 * w_2 + q_1 * w_1}{w_3} = \frac{8 * 3 + 4 * 6}{3} = 16\text{м/мин}$$

Условие $q_i \leq q_{max}$ выполняется, значение скорости людского потока берем из таблицы 5.1:

v_2 = 40м/мин;

Используя формулу (5.5) получаем время движения на третьем участке:

$t_3 = 8 / 40 = 0{,}2$ мин

На четвертом участке происходит слияние, и небольшое скопление людей, так как интенсивность на этом участке превышает максимальное значение:

$$q_4 = \frac{q_3 * w_3 + q_2 * w_2 + q_1 * w_1}{w_4} = \frac{16 * 3 + 8 * 3 + 4 * 6}{3} = 32\text{м/мин}$$

По таблице 5.1 принимаем значение предельной скорости при D = 0,9 м^2/ м^2:

$$v_4 = 15\text{м/мин};$$

Тогда время эвакуации на участке 4, на котором сливаются несколько потоков с учетом задержки движения в дверном проеме шириной $w_{дв}$ = 2м равно:

$$t_4 = \frac{l_4}{v_{пр}} + N * f\left(\frac{1}{q_{пр} * w_{дв}} - \frac{1}{\sum_{i=1}^{n} q_1 * w_1 + q_3 * w_3}\right) \qquad (5.7)$$

$$t_4 = \frac{3}{15} + 150 * 0{,}07\left(\frac{1}{8{,}5 * 2} - \frac{1}{4 * 6 + 8 * 3 + 16 * 3}\right) = 0{,}73\text{мин}$$

Время эвакуации по участкам 5,6,7 также затруднено, и вычисляется по формуле (5.7):

$$t_{5,6,7} = \frac{26}{8} + 150 * 0{,}07\left(\frac{1}{7{,}2 * 2} - \frac{1}{4 * 6 + 8 * 3 + 16 * 3 + 32 * 2}\right) = 3{,}9 \text{ мин}$$

Расчетное время эвакуации, определяемое по формуле (5.1) равно:

$$t_р = 0{,}08+0{,}1+0{,}2+0{,}72+3{,}9 = 5 \text{ мин.}$$

Время эвакуации с учетом задержек и скопления людей около дверных проемов и у выходов составило 5 минут. По межгосударственному стандарту (ГОСТ 12.1.004-91* «Пожарная безопасность») общее время эвакуации работников и учащихся из образовательного учреждения не должно превышать 6 минут.

Условие безопасной эвакуации полностью соблюдено.

5.3 Полная схема движения при эвакуации

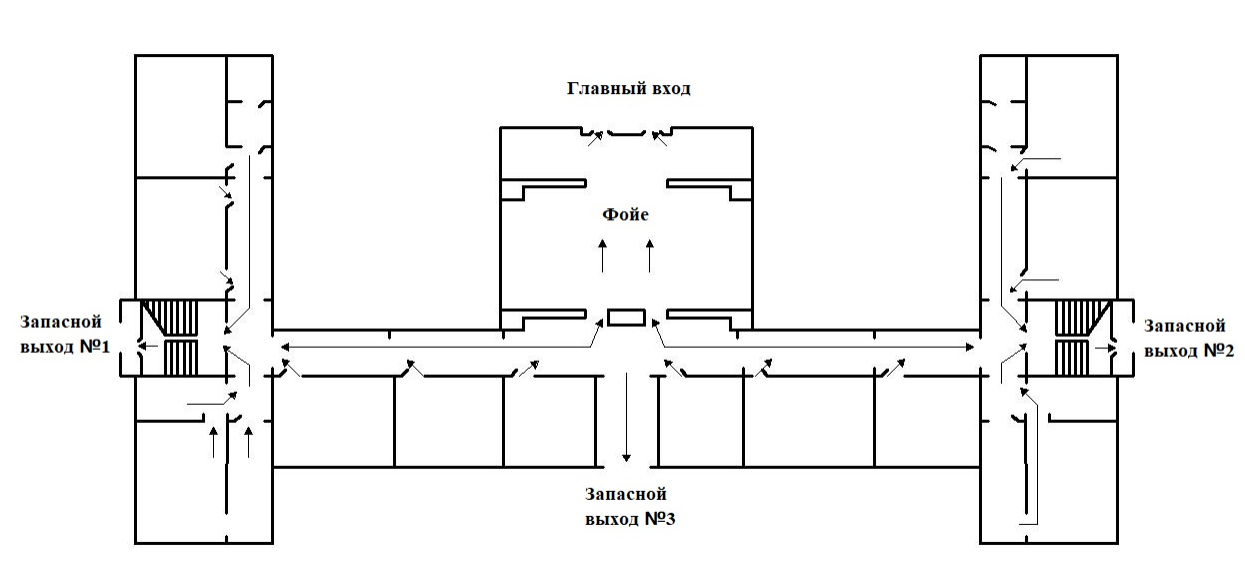


Рисунок 5.2 – Схема движения при эвакуации первого этажа.

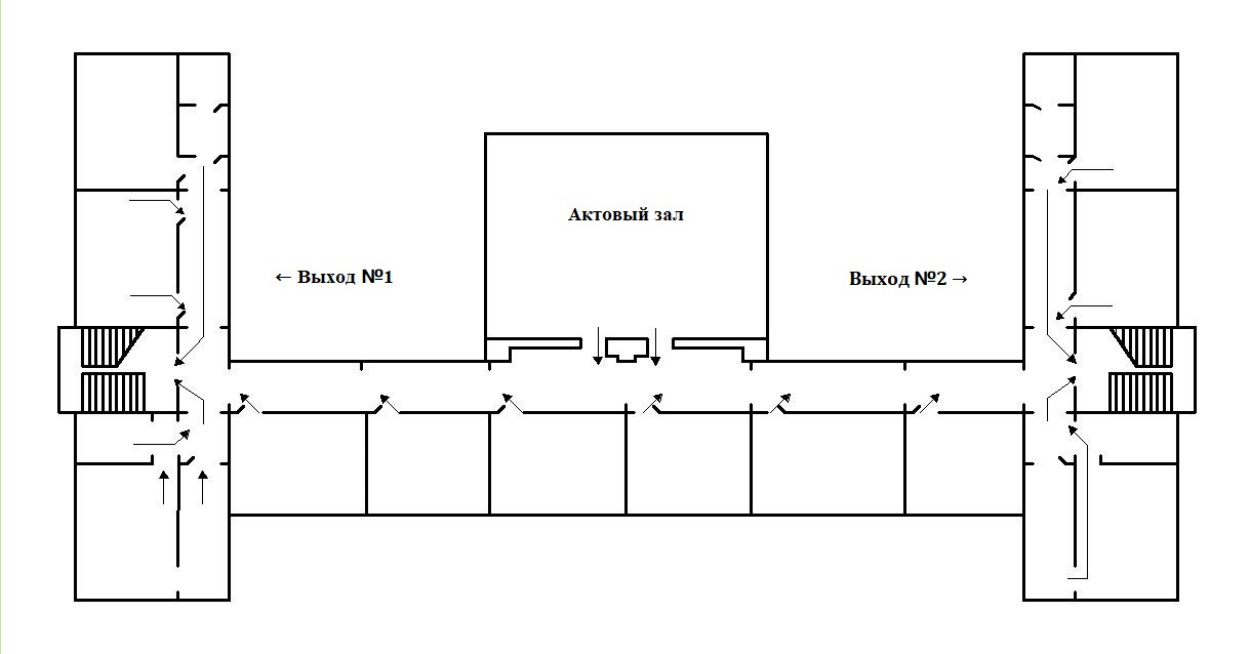
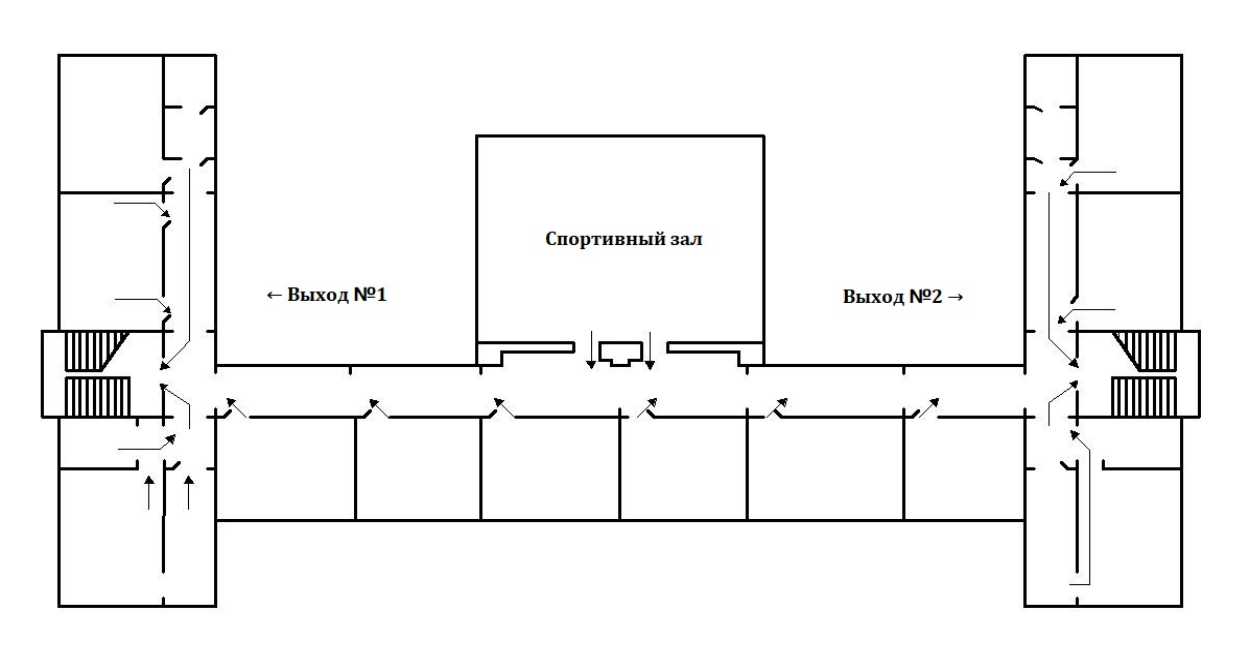


Рисунок 5.3 – Схема движения при эвакуации второго этажа.

Рисунок 5.4 – Схема дви



жения при эвакуации третьего этажа.

5.4 Вывод по разделу безопасность жизнедеятельности

В данном разделе дипломной работы было получено в ходе вычислений расчетное время эвакуации из коммунального государственного учреждения гимназии № 105. Был составлен и продуман план эвакуации из основного здания, с учетом одновременно находящихся учеников одной из смен, а также преподавательского состава.

Не стоит забывать, что выбраться из здания безусловно важно, однако это не единственное, что важно в экстренных ситуациях. Своевременное реагирование, оперативные, скоординированные и уверенные действия смогут спасти множество жизней, сэкономить драгоценные секунды при экстренной ситуации.

Заключение

В процессе работы над данным дипломным проектом, был проведен анализ процесса сбора информации в КГУ Гимназии №105, в результате которого было выявлено нерациональное использование людских, а именно преподавательских ресурсов для сбора информации. Для улучшения работы качества преподавателей, сокращения времени сбора и всеобщего удобства была создана база данных для Гимназии №105. В процессе разработки были изучены различные программные средства реализации базы данных, для построения приложения на базе платформы Android, и из них были выбраны наиболее удобные и совместимые между собой компоненты, такие как MySQL Database, Android studio, который включает в себя разработку на высокоуровневом и наиболее популяризированном языке программирования – Java.

В результате выполнения дипломного проекта, были получены мобильное приложение с простым, удобным и “user-friendly” интерфейсом, с возможностью быстрого доступа, а также расширения базы данных.

Себестоимость работы составила 2 124 804 тенге. Цена реализации программного продукта с учетом НДС составляет 2 855 737тг. Наибольшую часть от себестоимости разработки составило – машинное время – 71%. Аналогов данного программного продукта на рынке не представлено, с его внедрением эффективность сбора информации возрастет в сотни раз, что безусловно доказывает о необходимости выхода данного продукта на рынок.

Также в разделе безопасности жизнедеятельности был проведен анализ условий труда и произведен расчет времени эвакуации для КГУ Гимназии №105 и составлены планы эвакуации для основного здания учебного заведения, для своевременной и безопасной эвакуации персонала и детей.

Список литературы

1 Экономика и организация производства. Конспект лекций. - АУЭС, 2014.2. Боканова Г. Ш., Еркешева З.Д. Экономика и организация производства: Методические указания к выполнению расчетно-графических работ – АУЭС

2 Орлов Г.Г. Инженерные решения по охране труда в строительстве. Справочник строителя (1985), коричневое издение, под редакцией г. г.Оорлова, Москва, строииздат 1985

3 Распределенная обработка данных: курс лекций / Сост. Найханова Л.В. – Улан-Удэ, Издательство ВСГТУ, 2001. – 122 с.

4 Шилдт Г. Java, Полное руководство. Издательство Вильямс, 8-е издание, 2012.

5 Шилдт Г. Java, Руководство для начинающих. 5-е издание. Издательство Вильямс, 2012. – 624 с.

6 С.С.Тимофеева, В.В.Малов – Иркутск: Изд–во ИрГТУ, 2014. – 71 с, Расчет и проектирование систем обеспечения безопасности

7 https://metanit.com/java/tutorial/1.1.php

8 https://developer.android.com/guide/components/activities?hl=ru

9 http://www.webmasterwiki.ru/mysql

10 http://www.yaklass.ru/materiali?chtid=511&mode=cht

11 http://www.codenet.ru/db/interbase/ibsql/objs.php

12 https://webonto.ru/klassifikatsiya-baz-dannyih/

13 https://java.com/ru/about/

14 http://progaprosto.ru/doc/yazyk_programmirovaniya_java.php