Муниципальное бюджетное общеобразовательное учреждение

«Средняя общеобразовательная школа №4»

**Индивидуальный проект**

Разработка программы для автоматизации отчётностей классных руководителей

Работу выполнил:

Галкин Захар Васильевич,

ученик 11 «А» класса

Руководитель:

Уварова Виктория Ивановна,

учитель информатики

Мытищи

2023

**Содержание**

[**Введение** 3](#_Toc125988983)

[**1. Теоретическая часть** 4](#_Toc125988984)

[**1.1. Способы автоматизации отчётности** 4](#_Toc125988985)

[**1.2. Готовые решения для отчётности учителей** 4](#_Toc125988986)

[**2. Практическая часть** 5](#_Toc125988987)

[**2.1. Опрос учителей** 5](#_Toc125988988)

[**2.2. Информационная модель отчёта** 6](#_Toc125988989)

[**2.3. Создание макета программы** 8](#_Toc125988990)

[**2.4. Вёрстка приложения** 9](#_Toc125988991)

[**2.5. Серверная часть приложения** 10](#_Toc125988992)

[**Заключение** 11](#_Toc125988993)

[**Список литературы** 12](#_Toc125988994)

[**Приложения** 13](#_Toc125988995)

# **Введение**

Одним из наиболее важных социальных институтов в наше время является школа: опыт и умения, приобретаемые обучающимся в ней, помогают ему войти подготовленным в мир информации. Поэтому возможность наставника – преподавателя – качественно преподносить материал находится в приоритете.

возложив груз рутинной работы на автоматику.

**Актуальность:** в настоящее время образование

**Проблема** проекта заключается в том, что учителя выполняют отчётности, которые могут мешать преподаванию, но которые можно автоматизировать, предоставив монотонную работу компьютеру.

**Цель проекта:** создать приложение под Android, которое позволит классным руководителям автоматически собирать данные и составлять отчётность по ним.

**Задачи проекта:**

* Провести опрос классных руководителей с целью нахождения отчётов и документов, составление которых можно автоматизировать
* Составить информационную модель отчётов
* Разработать макет приложения в среде Figma
* Сверстать приложение по макету с использованием Jetpack Compose
* Создать серверную часть программы с базой данных

**Практическая значимость проекта** для классных руководителей, пользующихся данным приложением, могут получить все необходимые отчётности практически без временных затрат, имея обобщённую базу данных.

**Целевая группа:** классные руководители школ

# **1. Теоретическая часть**

## **1.1. Способы автоматизации отчётности**

## **1.2. Готовые решения для отчётности учителей**

<https://nsportal.ru/shkola/raznoe/library/2017/01/18/elektronnye-formy-otcheta-uchitelya>

<https://easyen.ru/load/metodika/programmy/ehlektronnyj_otchjot_uchitelja_predmetnika/276-1-0-13792>

https://pedsovet.su/load/48-1-0-25700

# **2. Практическая часть**

## **2.1. Опрос учителей**

## **2.2. Информационная модель отчёта**

Итак, мы выяснили, что наиболее востребованным видом отчёта для классных руководителей является социальный паспорт. Для дальнейшей работы необходимо чётко понять, с какой информацией придётся иметь дело. Или, выражаясь терминами, составить информационную модель социального паспорта. Это важно для однозначного понимания того, как представлять и обрабатывать те или иные данные.

Для составления информационной модели социального паспорта нужно выяснить, какие данные наиболее важны. Некоторую их часть, возможно, нашедшую бы применение в других областях, можно опустить ввиду её отсутствия в отчёте.

В социальный паспорт входит различная информация: данные непосредственно о самом классе, его составе, а также об обучающихся и их семьях. Для простоты восприятия можно выделить 3 основных сущности:

* Информация о человеке (HumanInfo)
* Информация о семье (FamilyInfo)
* Информация о классе (ClassInfo)

В сущность HumanInfo (информация о человеке) можно включить все данные, важные **в контексте социального паспорта**, а именно:

* ФИО
* Дата рождения
* Пол
* Контакты

Этот перечень необходимо дополнить данными, зависящими от того, является ли человек обучающимся, учителем или родителем. Для детей такой информацией будет служить список статусов и место учёбы TODO Приложение здесь. У родителей же, в свою очередь, это данные об образовании и наличии судимостей. Некоторые поля социального паспорта требуют информации о регистрации и документе, удостоверяющем личность, а потому нужно добавить и их.

Отдельно стоит упомянуть, что данные об отдельном человеке (HumanInfo) может как содержать, так и не содержать его учётную запись. В неё входят произвольный логин и пароль.

Следующей подмоделью социального паспорта является сущность, представляющая собой информацию о семье обучающегося (FamilyInfo). Помимо состава она должна содержать и данные о структуре, жилищных условиях и статусе семьи TODO приложение сюда.

И последней сущностью является информация о самом классе, для которого делается социальный паспорт. Здесь, помимо номера, классного руководителя и списка обучающихся, ничего не больше не предусмотрено.

Итак, все сущности описаны, и можно приступить к реализации информационной модели социального паспорта. Для этого я собираюсь использовать язык программирования Rust. Для выражения полей информационной модели можно использовать структуры языка:

* String – для описания текстовой информации
* Date – для описания дат
* Enum – для перечислений TODO приложение сюда
* i32 – для ID
* BTreeSet – для множеств (списков уникальных записей)

Основой реализации модели в программировании является композиция (объединение) нескольких небольших структур (struct в Rust) в большую структуру. Так и здесь, к примеру, структура FullName (ФИО) – это композиция из трёх String – строковых полей. Так нужно проделать со всем перечнем данных. Итого вышло 25 файлов, описывающих нашу информационную модель TODO приложение сюда.

## **2.3. Создание макета программы**

Информационная модель социального паспорта составлена, и теперь перед нами стоит задача иного плана: создать визуальный макет программы. Он нужен нам для понимания структуры и иерархии нашего приложения. В этом визуальный макет схож с информационной моделью: они дают представление о том, как всё должно быть расположено и взаимосвязано.

Надо сказать, что информационная модель уже находит себе применение: макет программы должен содержать формы для ввода определённых данных, и их последовательность играет большую роль в практичности программы и её эстетическом восприятии. Заранее выверенная структура позволяет объединять поля наиболее удобным для конечного пользователя способом.

Поскольку сразу после создания макета идёт верстка, он должен отражать структуру, композицию (объединение) полей в формы для ввода. А также задать определённый стиль приложению. Получается так, что макет нашей программы должен удовлетворять не одному, а нескольким требованиям:

1. Иерархия визуальных элементов наглядно показана
2. Общий стиль программы задан
3. Размеры текста и визуальных элементов согласованы

Для создания макета удобнее всего использовать виртуальные «доски». Такой функционал имеется у платформы под названием Figma, она и будет использоваться для наших целей.

Написать про удобство повторного использования элементов

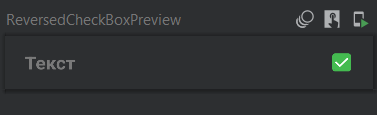
## **2.4. Вёрстка приложения**

После завершения работы над макетом программы можно непосредственно приступать к вёрстке приложения под Android. Для этого я буду использовать современный набор инструментов и библиотек – Jetpack Compose от компании Google.

Учитывая приобретённый во время создания макета опыт, среди общей массы визуального интерфейса TODO (возможно нужна отсылка на понятие) сразу можно выделить повторяющиеся элементы: кнопки, поля для текста и дат, иконки. Все простые составляющие экрана я решил вынести в отдельную группу примитивов. В дальнейшем эти элементы позволят более быстро верстать формы.

К числу примитивов относятся:

* Кнопки
* Поля для текста
* Поля для дат
* Иконки
* Выпадающий список
* Шаблон поверхности формы

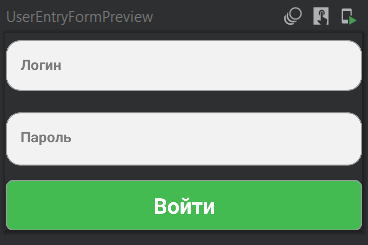


TODO (Добавить номер рисунка)

В конечном счёте вышло 15 визуальных элементов, относящихся к группе примитивов TODO (Ссылка на приложение).

Следующими после примитивных элементов идут формы. Они выполняют роль блока данных, в котором при нажатии на кнопку отправляется запрос на сервер с целью обновления информации. Для составления форм я применяю готовые примитивы, дело остаётся лишь за правильным их сочетанием.

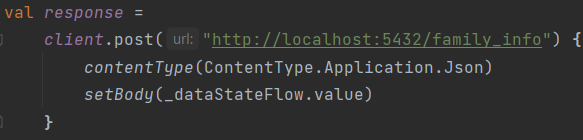
Каждая форма, в отличие от примитивов, привязана к определённой структуре из информационной модели, а потому напрямую взаимодействует с сервером. Так, например, функция UserEntryForm, являющая собой представление входа пользователя в систему, двусторонним образом согласуется с данными конкретного объекта UserInfo на серверной части.



TODO (Сделать нумерацию рисунка)

По завершении вёрстки всех форм, а именно, визуальной части без работы с данными, нужно позаботиться об их отправлении на сервер и дальнейшего их получения. Для этих целей было решено использовать библиотеку Ktor для языка программирования Kotlin. Она предоставляет удобный набор инструментов для создания серверных и клиентских сторон приложений.

Основу всего взаимодействия между моими формами и базой данных составили POST-запросы. Они позволяют изменять и обновлять информацию, необходимо лишь указать поля, которые нужно редактировать. Логику работы с запросами для каждого визуального элемента я переместил в отдельный тип – модель представления. Она обращается к серверу каждый раз, когда привязанная к ней форма вызывает событие – у нас это нажатие на кнопку.



TODO (Нумерацию в студию)

Таким образом и выстраивается всё взаимодействие серверной и клиентской частей. Итак, все формы готовы TODO (Приложение сюда).

Следующим этапом является вёрстка основных компонентов любого приложения – экранов. Они же, в свою очередь, делятся на страницы, по которым можно перемещаться с помощью навигационного меню. Для каждого экрана оно должно быть своё из-за разного содержания. Однако, есть и общие элементы, которые могут быть использованы повторно. Например, «шапка» приложения как на рисунке ниже:



TODO (Нумерация)

В содержимое страницы не должно входить ничего помимо сочетания форм. Именно в них и заключена вся логика работы с данными. После вёрстки всех частей можно приступить к экранам, имеющим один шаблон: сверху находится «шапка» приложения, снизу – навигационное меню, всё остальное занимают страницы. Для перемещения между визуальными элементами я использовал встроенную в Jetpack Compose библиотеку navigation-compose. Итого вышло TODO (число экр, стр, приложение со всеми страницами и экр)

## **2.5. Серверная часть приложения**

# **Заключение**

# **Список литературы**

# **Приложения**

