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](#_Toc126319221)

[**1. Теоретическая часть** 4](#_Toc126319222)

[**1.1. Способы автоматизации отчётности** 4](#_Toc126319223)

[**1.2. Готовые решения для отчётности учителей** 5](#_Toc126319224)

[**2. Практическая часть** 6](#_Toc126319225)

[**2.1. Информационная модель отчёта** 6](#_Toc126319226)

[**2.2. Создание макета программы** 9](#_Toc126319227)

[**2.3. Вёрстка приложения** 10](#_Toc126319228)

[**2.4. Создание модуля для автоматизации отчётности** 13](#_Toc126319229)

[**Заключение** 14](#_Toc126319230)

[**Список литературы** 15](#_Toc126319231)

[**Приложения** 16](#_Toc126319232)

# **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. Учитель может уделять большее количество сил на творческую составляющую работы, если возложить груз рутинных обязанностей на автоматику. В этом и заключается **актуальность** проекта.

Лично для меня эта тема представляет **интерес**, так как процесс изучения и систематизации информации, создания программы для автоматизации отчётности приносит мне настоящее удовольствие. К тому же, это замечательный опыт в программировании, который учитывается работодателями и приносит пользу классным руководителям.

Идея автоматизации отчётности далеко не нова, и уже придумано внушающее число решений для этого. В том числе, для преподавателей и классных руководителей. Однако не все реализации являются удобными и понятными для их пользователей, и то, что должно было упростить жизнь, лишь усложнило её, превратив сдачу доклада в сущий кошмар.

**Проблема** проекта заключается в том, что готовые на данный момент средства для автоматизации отчётности неудобны и сложны для понимания; больше внимания в них уделяется функциональности, а не практичности и удобству.

На фоне этого может выгодно выделиться моё решение проблемы – комфортная для использования программа для автоматизации отчётностей с современным дизайном – вот в чём **новизна** моей работы.

Итак, **целью проекта** является создание удобной программы, которая позволит классным руководителям хранить систематизированные данные и автоматически составлять отчётность по ним.

**Задачи проекта:**

* Изучить различные решения для автоматизации отчётности
* Составить информационную модель социального паспорта
* Разработать макет программы
* Сверстать программу по макету
* Создать модуль для автоматизации отчётности

**Используемые методы:** сбор информации, работа с ресурсами сети Интернет, систематизация и обобщение данных, моделирование, визуализация**,** программирование.

**Целевая аудитория:** классные руководители в школах

**Практическая значимость проекта** дляклассных руководителей, пользующихся данным приложением, – это возможность получить необходимые отчётности гораздо быстрее, чем если это делать вручную.

# **1. Теоретическая часть**

## **1.1. Способы автоматизации отчётности**

Как уже говорилось во введении, существует множество способов создания

<https://vc.ru/services/151096-volshebnaya-palochka-avtomatizacii-kak-delat-otchety-s-udovolstviem>

<https://aif.ru/boostbook/avtomatizatsija-ucheta.html#complex>

Рассказать про Excel, CRM

## **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>

1C

# **2. Практическая часть**

## **2.1. Составление информационной модели отчёта**

Одним из наиболее востребованных видов отчёта для классных руководителей является социальный паспорт. Он позволяет наглядно увидеть возможные причины неуспеваемости обучающихся, но его составление занимает большое количество времени. Именно социальный паспорт я и решил взять за основу. 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.

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.

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 Kotlin. Для выражения полей информационной модели можно использовать структуры и объекты языка:

* String – для описания текстовой информации
* Date – для описания дат
* object – для перечислений TODO приложение сюда
* Int – для ID
* Set – для множеств (списков уникальных записей)

Основой реализации модели в программировании является композиция (объединение) нескольких небольших структур (data class в Kotlin) 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.

## **2.2. Создание дизайна программы**

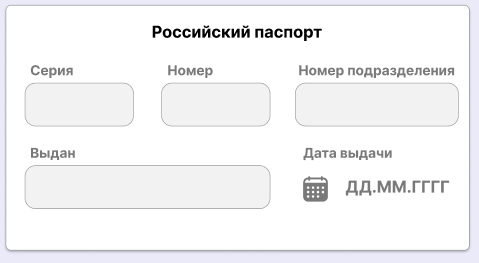
Информационная модель социального паспорта составлена, и теперь перед нами стоит задача иного плана: создать современный дизайн для Android-приложения. Это важный этап работы: мне не хотелось бы повторить судьбу громоздких, неудобных программ и таблиц. За основу я решил взять принципы такого направления дизайна, как Material Design TODO (Ссылку сюда), подобрал палитру цветов и определил типографию.

Дизайн программы нужен нам для понимания структуры, стиля и иерархии нашего приложения, поскольку сразу после создания дизайна идёт вёрстка. Получается так, что **визуальный** **макет** нашей программы **должен удовлетворять** нескольким **требованиям:**

1. Иерархия визуальных элементов наглядно показана
2. Общий стиль программы задан
3. Размеры текста и визуальных элементов согласованы
4. Интерфейс интуитивно понятен для пользователя

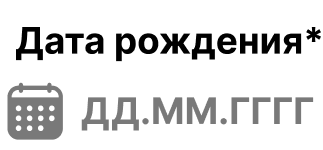
Создание макета будет удобнее всего проходить в виртуальных «досках». Их функционал имеется у платформы под названием Figma, она и будет использоваться для наших целей. Ключевой особенностью данного инструмента является возможность объединения визуальных простых элементов в группы – так создаются более сложные части интерфейса. Позднее, в процессе вёрстки, это сыграет нам на руку.

Для каждой сущности из информационной модели я выделил отдельный визуальный элемент: поле или форму для ввода определённых данных. Заранее выверенная структура позволяет объединять поля так, чтобы конечному пользователю было удобно работать в приложении:



TODO нумерацию сюда

Для улучшения интуитивности будущей программы я применял небольшие минималистичные картинки – иконки. Они являются хорошим дополнением к визуальному интерфейсу:

TODO нумерацию сюда TODO и сюда нумерацию

После завершения работы над полями и формами для ввода данных из информационной модели нужно приступить к созданию элементов самого приложения, не относящихся к самой информации отчёта. К их категории я отнёс кнопки, навигационные меню TODO (приложение сюда), верхнюю панель и некоторое другое. Все они в совокупности с формами составят экраны программы TODO приложение сюда. Для создания их макета я скомпоновал все готовые визуальные элементы воедино; многократное переиспользование гораздо ускорило этот процесс. Итого, получился следующий дизайн TODO (приложение сюда). Ссылка на макет ниже: <https://www.figma.com/file/dTYBakzwC9Xjz5DdDsZjP2/My-teacher?node-id=0%3A1&t=XluDVDpCfJr2fch4-0> TODO приложение с QR

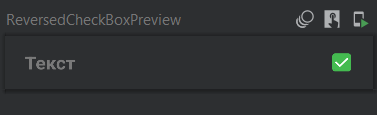
## **2.3. Вёрстка приложения**

После завершения работы над дизайном 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

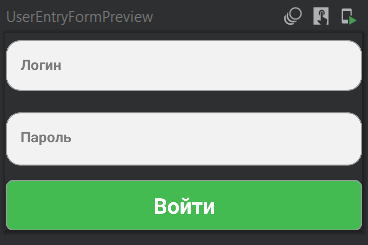


TODO (Добавить номер рисунка)

В конечном счёте вышло TODO визуальных элементов, относящихся к группе примитивов TODO (Ссылка на приложение). Теперь они представляют собой наименьшую единицу пользовательского интерфейса, всё остальное будет являться сочетанием примитивов.

Следующими после примитивных элементов идут формы. Они выполняют роль блока данных, которые сохраняются в локальное хранилище при нажатии на кнопку. Для составления форм я применил готовые примитивы, и, как было сказано ранее, дело остаётся лишь за правильной их композицией.

Каждая форма, в отличие от примитивов, привязана к определённой структуре из информационной модели, а потому взаимодействует с данными хранилища через посредника. TODO (Сделать фото формы социального паспорта, добавить описание) Так, например, функция UserEntryForm, являющая собой представление входа пользователя в систему, двусторонним образом согласуется с информацией конкретного объекта UserInfo базы данных через модель представления.



TODO (Сделать нумерацию рисунка)

По завершении вёрстки всех форм, а именно, визуальной части без работы с данными, нужно позаботиться об их отправлении в локальное хранилище и дальнейшего их получения. Для этих целей было решено использовать библиотеку Room Database для языка программирования Kotlin. Она предоставляет удобный набор инструментов для работы с базами данных.

Основу всего взаимодействия между моими формами и базой данных составили SQL-запросы. Они позволяют получать, изменять и обновлять информацию, необходимо лишь указать поля, которые нужно редактировать. Логику работы с запросами для каждого визуального элемента я переместил в отдельный тип – репозиторий. Он обращается к базе данных каждый раз, когда модель представления передаёт событие формы – у нас это нажатие на кнопку. Таким образом и выстраивается всё взаимодействие хранилища с визуальным представлением.

Итак, все формы готовы TODO (Приложение сюда). Из них будут складываться уже более крупные элементы пользовательского интерфейса.

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 одной формы, в которой и заключена вся логика работы с данными и назначение страницы, и функции навигации.

На этом этапе работы я столкнулся с проблемой: объём вёрстки, заданный макетом программы – почти пара десятков страниц у нескольких экранов – оказался слишком большим. Помимо этого, возникли и технические трудности с компоновкой элементов. Поэтому я принял решение сократить содержимое программы до рабочего минимума, убрать интересные, но второстепенные особенности. После сжатия осталось всего три экрана: для человека, семьи и класса. Это был горький, но поучительный опыт.

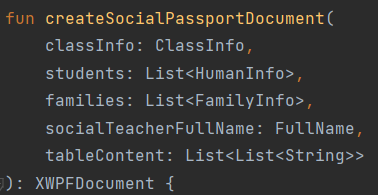
После вёрстки всех оставшихся после сокращения частей я приступил к экранам, имеющим один шаблон: сверху находится «шапка» приложения, снизу – навигационное меню, всё остальное занимают страницы TODO (экран с подписанными частями). Для перемещения между визуальными элементами я использовал встроенную в Jetpack Compose библиотеку navigation-compose. Вся вёрстка экранов свелась к созданию меню и расположению уже страниц. Итого вышло четыре экрана с общим числом страниц в 11 единиц TODO (число экр, стр, приложение со всеми страницами и экр).

## **2.4. Создание модуля для автоматизации отчётности**

Итак, само приложение готово – дело осталось за малым. Нужно создать модуль для автоматического составления отчёта по социальному паспорту. Конечным результатом будет документ Microsoft Word. У меня есть шаблон социального паспорта, сохранённый в формате этой программы, а потому достаточно только воссоздать его копию в коде языка программирования и подставить необходимые данные. Создавать документы Word нужно под платформу Android. Необходимый нам набор инструментов предлагает библиотека Apache POI для языка программирования Kotlin.

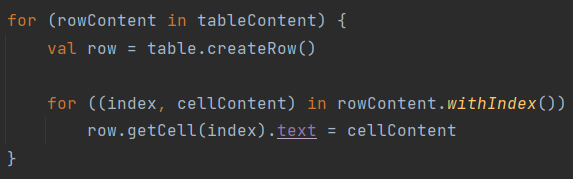
Перед тем как начать делать сам документ, нужно подготовить данные для параграфов и таблицы, имеющей 5 столбцов и 26 строк: информацию о классе, обучающихся и семьях

Таблица содержит данные о семьях и учениках, имеющих определённые статусы, а потому лучше сделать функцию, возвращающую данные в виде матрицы строк (List<List<String>> в Kotlin). При наличии статуса у ребёнка или семьи подставляются данные в формате: Статус – ФИО детей – Дата рождения и место учёбы – ФИО родителей и номер документа – Домашний адрес и контакты. И так для каждого ряда таблицы. Данные для параграфов представляют общие характеристики – число мальчиков и девочек в классе, число родителей с высшим образованием и так далее, а потому не нуждаются в отдельной функции. Итого, так выглядят данные для документа:



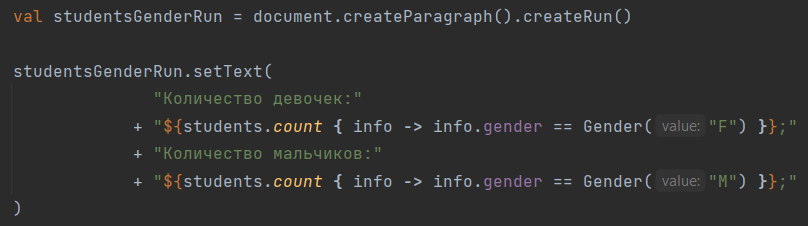
TODO (Нумерацию)

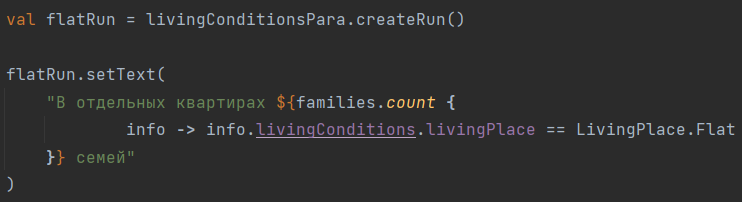
Когда данные для заполнения были готовы, я взялся за формирование документа. Заполнение таблицы в документе выглядит таким образом:



TODO (нумерацию)

Заполнение некоторых параграфов:





TODO (Нумерацию)

После завершения «вёрстки» документа его нужно сохранить. Для этого нужно создать файл со следующим названием:

«Социальный паспорт [Номер класса] от [Дата создания].docx»,

передать в файловый поток вывода (FileOutputStream) и вызвать метод

document.write(FileOutputStream(File(directory, «Социальный паспорт ${classInfo.Number} от ${date}.docx»,))). Отчёт готов!

# **Заключение**

# **Список литературы**

1. <https://habr.com/ru/company/redmadrobot/blog/252773/>
2. https://habr.com/ru/post/503444/

# **Приложения**