**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра Веб-технологий и компьютерного моделирования**

Кречко Екатерина Сергеевна

**РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ШАБЛОНА MVP**

Дипломная работа

Научный руководитель:

старший преподаватель

Блинов Игорь Николаевич

Допущен к защите

«\_\_\_» \_\_\_\_\_\_\_\_ 2017 г.

Зав. кафедрой Веб-технологий и компьютерного моделирования

кандидат физико-математических наук, доцент В.С. Романчик

Минск, 2017

**РЕФЕРАТ**

**РЭФЕРАТ**

**ABSTRACT**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc475642183)

[Глава 1. НЕОБХОДИМЫЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ 7](#_Toc475642184)

[1.1 Исследование предметной области 7](#_Toc475642185)

[1.2 Обзор существующих инструментов для достижения цели 7](#_Toc475642186)

[1.2.1 Общий обзор мобильных технологий на рынке 7](#_Toc475642187)

[1.2.2 Подведение итогов: типы и особенности мобильных приложений 7](#_Toc475642188)

[1.3 Основные шаблоны разработки для мобильных устройств 7](#_Toc475642189)

[1.3.1 Шаблон Model-View-Controller 7](#_Toc475642190)

[1.3.2 Шаблон Model-View-Presenter 7](#_Toc475642191)

[1.3.3 Шаблон Model-View-ViewModel 7](#_Toc475642192)

[1.3.4 Сравнение шаблонов 7](#_Toc475642193)

[Глава 1. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ 8](#_Toc475642194)

[1.1 Постановка задачи 8](#_Toc475642195)

[1.1 Разработка архитектуры приложения 8](#_Toc475642196)

[1.2 Разработка базового функционала приложения 8](#_Toc475642197)

[1.2.1 Стартовая страница 8](#_Toc475642198)

[1.2.2 Расписание студента 8](#_Toc475642199)

[1.2.3 Расписание преподавателя 8](#_Toc475642200)

[1.2.4 Детали занятия 8](#_Toc475642201)

[1.2.5 Добавление материалов 8](#_Toc475642202)

[1.2.6 Настройки 8](#_Toc475642203)

[Глава 2. ВОЗМОЖНЫЕ ПУТИ РАЗВИТИЯ ПРИЛОЖЕНИЯ 9](#_Toc475642204)

[1.1 Дальнейшие перспективы развития приложения 9](#_Toc475642205)

[ЗАКЛЮЧЕНИЕ 10](#_Toc475642206)

# ВВЕДЕНИЕ

*Во введении:*

* раскрывается значение избранной темы и проблем, рассматриваемых в работе, обосновываются актуальность темы, формулируются цель и задачи исследования, определяются хронологические рамки исследования и аргументируется правомерность их выбора (при необходимости);
* определяется необходимость проведения исследований по данной теме для решения конкретной проблемы;
* дается краткая экономическая характеристика организации, на материалах которой выполняется дипломная работа, если работа выполняется на примере конкретной организации или предприятия;
* указываются разделы, содержащие описание используемых методов и (или) методик, собственных теоретических и экспериментальных исследований, результаты расчетов и другие сведения, определенные заданием;
* приводится определение экономической эффективности внедрения полученных результатов, требования охраны труда и техники безопасности при эксплуатации

*Основная часть* дипломной работы содержит главы (разделы), каждая из которых включает не менее двух параграфов (параграфы могут быть разделены на пункты). Все главы (разделы (параграфы)) дипломной работы по объему должны быть примерно равнозначны.

**Первая глава** (раздел) независимо от структуры работы носит теоретико-методологический характер. В ней на основе изучения работ отечественных и зарубежных авторов излагается сущность исследуемой проблемы, рассматриваются различные подходы к ее решению, дается их оценка, обосновывается точка зрения автора работы. Дается характеристика основным методам исследования.

**Остальные главы** (разделы) дипломной работы, носят расчетно-аналитический (аналитический) характер. В них проводится глубокий анализ изучаемой проблемы с использованием различных методов исследования, дается технико-экономическая характеристика организации (предприятия, учреждения), на материалах которой выполняется дипломная работ, анализируются пакеты прикладных программ и пр. При этом автор дипломной работы не ограничивается констатацией фактов, а вскрывает недостатки и причины, их обусловившие, намечает пути их устранения.

Автором разрабатываются и обосновываются конкретные предложения по совершенствованию определенной сферы деятельности, направления и пути решения исследуемой проблемы. Для обоснования целесообразности предлагаемых мероприятий показывается, как их реализация отразится на общих показателях деятельности организации (отрасли), а также оценивается эффективность их внедрения при анализе деятельности конкретных предприятий и организаций.

Каждая глава (раздел) дипломной работы завершается краткими выводами, вытекающими из исследования.

# НЕОБХОДИМЫЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

## Исследование предметной области

В 2016 компания Google провела исследование, которое называлось «Особенности повеления интернет-пользователей». В опросах для данного исследования было опрошено около 60 тысяч человек из 61 страны, среди которых была и Беларусь. По результатам исследования 98% пользователей до 35 лет заходят в интернет ежедневно, 89% молодых пользователей используют для этого смартфоны.

Наиболее распространёнными являются смартфоны с операционными системами iOS и Android. По сведениям из различных источников, доля рынка смартфонов на Android занимает до 90% рынка Беларуси.- введение

Изначально операционная система Android разрабатывалась компанией Android Inc., но в 2005 году компания Google купила компанию, занимавшуюся разработкой операционной системы. Уже 23 сентября 2008 года официально вышла первая версия операционной системы и полноценный пакет разработчика Software Development Kit 1.0 (SDK 1.0). С тех пор компания Google продолжает непрерывную работу над разработкой системы и регулярно выпускает новый версии продукта.

Android – операционная система, разработанная на базе ядра Linux. Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android – открытая операционная система, которая предоставляет возможности интеграции с огромным и постоянно растущим числом сервисов от компании Google. Открытость операционной системы дает возможность настройки смартфона под определенного пользователя и его нужды.

## Обзор существующих инструментов для достижения цели

Android Studio – интегрированная среда разработки (IDE) для работы с платформой Android. Android Studio была разработана компанией Google на основе программного обеспечения Intellij IDEA от компании JetBrains. Первая версия 1.0 была выпущена в 2014 году, хотя среда разработки была доступна с версии 0.1, опубликованной в мае 2013 года. Android Studio является официальным средством разработки Android приложений и, так же как и операционная система, постоянно поддерживается и улучшается компанией Google.

Android Studio предоставляет множество встроенных функций, позволяющих максимизировать производительность разработчика и автоматизировать его рутинную работу, позволяя разрабатывать приложения максимально эффективно.

На рисунке ниже представлен главный экран Android Studio

При установке Android Studio устанавливается ряд дополнительных инструментов, таких как Android SDK и Android Virtual Device (AVD). Android Virtual Device – эмулятор реального девайса на Android. В зависимости от назначения разрабатываемого приложения можно создавать различный виртуальные устройства, от обычных смартфонов до телевизоров и умных часов. При создании устройства можно задать собственные характеристики, такие как диагональ экрана, его ширина и высота, объем памяти и версию операционной систем. Этот инструмент позволяет отлаживать и тестировать приложение, поддерживаемое на любом устройстве, вне зависимости от его отдельных характеристик.

AVD позволяет не только запускать приложения, но и полноценно работать с ними. Инструмент эмулирует все основные системы устройства: работа с виртуальной клавиатурой, работу с мобильной сетью и сетями Wi-Fi, работу с GPS навигацией, физическими датчиками и кнопками устройства.

Рисунок

## Основные шаблоны проектирования для мобильных устройств

Шаблон проектирования – некая архитектурная конструкция, представляющая собой решение проблемы, часто возникающей при разработке программного обеспечения.

### Обзор семейства шаблонов Model-View

Рассмотрим очевидный подход к разработке приложения с пользовательским интерфейсом. Разработчик построит форму, добавит на нее интерфейс, добавит туда код для отображения элементов управления, логику обработки событий интерфейса, данные и методы их обработки.

Такой прямолинейный подход хорош, но только для очень простых интерфейсов без сложной логики. При разработке приложений с более сложной логики у такого подхода сразу раскрывается ряд недостатков:

1. Невозможность повторного использования кода. Код данных, написанный в реализации интерфейса, можно использовать только в связке с этим интерфейсом. Логика, реализованная в интерфейсе, может быть использована только с этим интерфейсом.
2. Высокая сложность поддержки кода. Небольшое изменение в пользовательском интерфейсе повлечет за собой неоправданно большие изменения в остальных частях кода. С ростом приложения, числа его функций и элементов интерфейса сложность поддержки только растет.
3. Ухудшение тестируемости. Код не поддается модульному тестированию. В таком случае используется тестирование пользовательского тестирования. Для ручного тестирования – нужен конечный пользователь (тестировщик), для автоматизированного – большие временные затраты на разработку тестов и, возможно, специалиста по автоматизации тестов.

Основная причина появления всех проблем заключается в нарушении одного из принципов объектно-ориентированного программирования – принципа единственной ответственности. В соответствии с этим принципом, любой объект должен иметь лишь одну обязанность; у класса должна быть только одна причина для его изменения. При описанном подходе к разработке класс интерфейса отвечает и за изменения в представлении, и за логику, и за обработку данных.

MV\* шаблоны предназначены для разделения кода обработки данных (Model), кода пользовательского интерфейса (View) и кода логики (Presenter, Controller, ViewModel и др.). Такой подход имеет ряд преимуществ перед описанным прямолинейным подходом. Код становится более поддерживаемым, появляется возможность повторного использования и тестирования отдельных модулей. Такой код не нарушает принципов объектно-ориентированного программирования и, в частности, принципа единственной ответственности.

Основными представителями семейства MV\* шаблонов являются Model-View-Controller, Model-View-ViewModel, Model-View-Presenter. Рассмотрим подробнее общие черты этих шаблонов.

**Model.** Модель представляет собой данные для работы приложения. Модель данных должна быть полностью независима от всех остальных частей продукта. Одна модель может иметь множество различных представлений и контроллеров. Независимость модели дает возможность использовать ее повторно.

**View.** Представление отвечает за отображение данных, полученных от модели. Представление – пользовательский интерфейс, методы, отвечающие за его изменение. Классы представления содержат только логику, отвечающую за работу с интерфейсом пользователя.

[*https://habrahabr.ru/company/mobileup/blog/313538/*](https://habrahabr.ru/company/mobileup/blog/313538/)

### Шаблон Model-View-Controller

Пример с автомобилем? <http://www.phpinfo.su/articles/theory/model_view_controller.html>

Одним из первых шаблонов разделения представления от логики и модели стал шаблон Model-View-Controller. Концепция MVC была описана в конце 1970-х годов Трюгве Реенскаугом, работавшем над языком программирования «Smalltalk». Этот шаблон в корне изменил взгляд на проектирование приложений с пользовательским интерфейсом.

**Controller.** Контроллер представляет собой управляющую логику и отвечает за взаимодействие модели и представления. Контроллер направляет данные от пользователя к системе и обновляет модель или представление. Когда пользователь взаимодействует с приложением, контроллер получает уведомление о действиях пользователя и решает, что с ними делать.

Поскольку шаблон был разработан во времена, когда программирование кардинально отличалось от того, что мы знаем сейчас, нет строгих правил реализации шаблона MVC. Запросы к базам данных, валидация данных, реализация бизнес-логики может размещаться как в модели, так и в контроллере. В зависимости от размещения этих функций, модель может трактоваться по-разному. Различают *пассивную* и *активную* модели.

Пассивная модель не имеет никаких средств воздействия ни на представление, ни на контроллер. В таком случаем модель используется исключительно в качестве источника данных. Все изменения в модели отслеживаются контроллером, который обновляет представление.

Активная модель содержит не только данные, но и оповещает представление об изменениях. Модель при такой реализации содержит бизнес-логику приложения. Представления, в такой реализации, подписываются на изменения в модели. Такая реализация позволяет сохранить независимость модели как от контроллера, так и от представления.



Рисунок 1.1 Схема модели MVP – перерисовать!

### Шаблон Presentation Model

**PresentationModel.** Модель представления содержит логику пользовательского интерфейса.

Часто информация, содержащаяся в модели не может быть непосредственно использована на представлении. Модель представления может преобразовывать данные из модели, дополнять их или собирать данные из нескольких источников. Это наиболее вероятно, когда у вас нет полного контроля над моделью. Например, если вы получаете данные от сторонних веб-сервисов или же из базы данных существующего приложения.

Модель представления хранит состояние пользовательского интерфейса. В свойствах могут храниться данные, никак не связанные с моделью. Это может быть элемент, выбранный в данный момент; ошибки валидации данных.

Представление может извлекать данные из модели представления и получать всю необходимую информацию для отображения. Одно из преимуществ такого подхода заключается в том, что вы можете создать логическое и полностью тестируемое представление вашего UI, не полагаясь на тестирование визуальных элементов.

### Шаблон Model-View-ViewModel

Шаблон MVVM впервые был представлен Джоном Госсманом в 2005 году как модификация шаблона Presentation Model.

**ViewModel.**

### Шаблон Model-View-Presenter

<https://github.com/antoniolg/androidmvp>

<https://www.youtube.com/watch?v=oiNfPjV72lg>

<https://www.youtube.com/watch?v=2zpW2c5NkIQ>

MVP достаточно универсальный паттерн и подойдет во многих случаях (это мое личное мнение). Какой вариант использовать: Passive View или Supervising Controller – решать вам. Руководствуйтесь тем, что вам нужно: больше контроля и тестируемости либо лаконичности и краткости кода. Лавируйте между задачами и применяйте тот или другой подход.

### Сравнение шаблонов

Табличка с основными характеристиками шаблонов

# РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

## Постановка задачи

## Разработка архитектуры приложения

https://www.tutorialspoint.com/design\_pattern/mvc\_pattern.htm

## Разработка базового функционала приложения

### Стартовая страница

### Расписание студента

### Расписание преподавателя

### Детали занятия

### Добавление материалов

### Настройки

# ВОЗМОЖНЫЕ ПУТИ РАЗВИТИЯ ПРИЛОЖЕНИЯ

## Дальнейшие перспективы развития приложения

В ***заключении*** логически и последовательно излагаются теоретические и практические выводы по каждому разделу дипломной работы. Выводы и предложения должны быть конкретными, реальными и обоснованными, вытекать из результатов проведенного исследования и содержания дипломной работы. Выводы пишутся тезисно (по пунктам). Из каждого подраздела теоретической части рекомендуется в заключение включать не более одного вывода. Количество выводов по каждому подразделу расчетно-аналитической и конструктивной частей дипломной работы может быть более одного.

# ЗАКЛЮЧЕНИЕ

# СПИСОК ЛИТЕРАТУРЫ

1. https://dev.by/lenta/main/google-91-belorusov-ezhednevno-prisutstvuyut-onlayn