

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Мобильное приложение – агрегатор аптечных
товаров**

Студент	_____	Кулаева Ксения Николаевна
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	Балашова И.Ю.
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	Такташкин Д.В.
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	Козлов А.Ю.
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	Попова Н.А.
	(подпись)	(фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Кулаева Ксения Николаевна гр. 19ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Мобильное приложение – агрегатор аптечных товаров

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

2. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Балашова И.Ю.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: мобильное приложение, обеспечивающее сравнение стоимости
лекарств в различных аптеках города Пенза.

2) Среда выполнения приложения: операционная система Android версии 6.0 и выше
Языковые средства разработки: C#, T-SQL, UML.

Инструментальные средства разработки: Microsoft SQL Server 2018, Microsoft
Visual Studio 2022, Microsoft Project 2007, StarUML, Visual Paradigm 17.0

Технологии разработки: ООП, RUP

3) Основные функции мобильного приложения:

а) авторизация и регистрация;

б) просмотр списка аптечных товаров;

в) поиск товаров внутри категорий и во всем списке;

г) осуществление заказа;

д) выбор темы интерфейса.

4. Перечень подлежащих разработке вопросов

<i>1) Постановка задачи и анализ предметной области</i>
<i>2) Проектирование</i>
<i>3) Реализация</i>
<i>4) Контроль качества программного обеспечения</i>
<i>5) План разработки и оценка бюджета</i>
<i>6) Оформление пояснительной записки</i>

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 114 листов, 59 рисунков, 20 таблиц, 21 использованный источник, 6 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, АГРЕГАТОР АПТЕЧНЫХ ТОВАРОВ, ANDROID, C#, XAMARIN, UML, БАЗА ДАННЫХ, MICROSOFT VISUAL STUDIO, MICROSOFT SQL SERVER

Объектом разработки является средство автоматизации предоставления фармацевтических услуг населению.

Цель работы – разработка приложения – агрегатора, которое позволит сгруппировать товары из разных аптек в одном месте.

Разработка базы данных проводилась в MICROSOFT SQL SERVER 2018 на языке T-SQL.

Разработка приложения проводилась на языке C# в среде разработки Microsoft Visual Studio 2022 в операционной системе Windows 10. Приложение предназначено для использования на устройствах под управлением операционной системы Android версии 5.0 и выше.

					ПГУ 09.03.04 – 08BP191.11 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Мобильное приложение – агрегатор аптечных товаров. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Кулаева К.Н.					4	114
Пров.		Балашова И.Ю.				Группа 19ВП1		
Реценз.								
Н. контр.		Такташкин Д.В						
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ мобильных приложений-агрегатов для аптечных сетей	9
1.1 Анализ предметной области и постановка задачи.....	9
1.2 Обзор и анализ программных средств-агрегаторов аптечных товаров.....	9
1.2.1 Мобильное приложение «Аптека Вита экспресс».....	9
1.2.2 Мобильное приложение «Аптека.ру»	10
1.2.3 Мобильное приложение «Еаптека».....	12
1.2.4 Мобильное приложение «Все аптеки».....	13
1.2.5 Мобильное приложение «Здравсити».....	14
1.2.6 Мобильное приложение «Мегааптека»	15
1.2.7 Сравнение мобильных приложений-агрегаторов аптечных товаров	16
1.3 Анализ функциональных требований	18
1.4 Анализ нефункциональных требований	21
2 Проектирование мобильного приложения-агрегатора аптечных товаров...	22
2.1 Архитектура приложения.....	22
2.2 Проектирование структур данных	22
2.2.1 Концептуальная модель базы данных.....	23
2.2.2 Логическая модель базы данных	24
2.3 Проектирование пользовательского интерфейса.....	24
2.4 Проектирование программных средств	29
3 Реализация мобильного приложения-агрегатора аптечных товаров.....	31
3.1 Реализация базы данных.....	31
3.2 Наполнение базы тестовыми данными	31
3.3 Реализация хранимых процедур и функций.....	33
3.4 Реализация клиентской части приложения	34
3.5 Реализация интерфейса приложения	36
4 Контроль качества программного обеспечения.....	41
4.1 Выбор режима тестирования	41

4.2 Функциональная диаграмма и таблица решений.....	42
4.3 Тестирование интерфейса приложения	45
4.4 Метрики кода.....	46
4.4.1 Метрика SLOC.....	46
4.4.2 Цикломатическая сложность	48
4.4.3 Метрика Холстеда.....	49
4.4.4 Метрика Чепина	51
4.5 Планирование разработки и оценка бюджета.....	52
Заключение	54
Список использованных источников	55
Приложение А. Атрибуты сущностей базы данных.....	57
Приложение Б. Графическая часть.....	61
Приложение В. Скрипт для создания базы данных.....	66
Приложение Г. Скрипты хранимых процедур	69
Приложение Д. Листинг мобильного приложения.....	72
Приложение Е. Результаты тестирования приложения	104

Введение

Сейчас в городе Пенза работает большое количество аптечных сетей. В каждой из них свой ассортимент продаваемых товаров и различные цены на них.

Для покупателя важно купить интересующий его товар по выгодной цене. Ему неудобно ходить от одной аптеки до другой, сравнивая ассортимент.

Почти у каждой крупной аптеки есть свое мобильное приложение. С его помощью можно узнать стоимость товара, не выходя из дома. Имея на своем смартфоне несколько таких приложений, можно сравнивать цены в различных аптечных сетях. Однако переключение между приложениями занимает время, также подобный способ сравнения не является наглядным и удобным для пользователя. Кроме того, десяток приложений с аналогичным функционалом занимает много места в памяти смартфона.

Из вышесказанного очевидно, что существует необходимость в мобильном приложении-агрегаторе аптечных товаров, которое будет поддерживать возможность поиска и заказа лекарств из различных аптек города и их сравнение на одном экране.

Объект исследования – средства автоматизации предоставления фармацевтических услуг населению.

Предмет исследования – информационная система, осуществляющая оформление заказов из нескольких аптек города Пенза, состоящая из базы данных и мобильного приложения.

Целью выпускной квалификационной работы является разработка приложения – агрегатора, которое позволит сгруппировать товары из разных аптек в одном месте. Разработка включает в себя анализ требований, проектирование, реализацию и тестирование программного обеспечения.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести сравнительный анализ программных продуктов-аналогов и выделить основные результаты;

- провести анализ требований к разрабатываемому программному продукту;
- осуществить проектирование программных средств и структур данных разрабатываемого программного продукта;
- провести планирование и расчет бюджета разрабатываемого программного продукта;
- осуществить разработку программного продукта;
- провести тестирование разработанного программного средства;
- оформить пояснительную записку в соответствии с требованиями.

В работе использованы методы сравнительного анализа, методы проектирования и разработки программного обеспечения.

1 Анализ мобильных приложений-агрегатов для аптечных сетей

1.1 Анализ предметной области и постановка задачи

Аптека – особая специализированная организация системы здравоохранения, занимающаяся изготовлением, фасовкой, анализом и продажей лекарственных средств.

По характеру деятельности аптеки можно классифицировать на два типа: производственные, где помимо торгового зала находятся производственные помещения, и аптеки готовых лекарственных форм, которые занимаются только продажей лекарств, изготовленных на заводах фирм-производителей. В рамках данной работы будут рассматриваться последние.

Для определения требований к разрабатываемому приложению необходимо провести анализ предметной области и выявить наиболее вероятные варианты использования.

Данное приложение разрабатывается для одной категории пользователей – покупателей аптечных товаров.

Покупатель будет иметь возможность авторизоваться в приложении, получить одновременный доступ к ассортименту нескольких аптек, просматривать список аптечных товаров в данных аптеках, осуществлять поиск товаров по их названию и сделать заказ.

1.2 Обзор и анализ программных средств-агрегаторов аптечных товаров

1.2.1 Мобильное приложение «Аптека Вита экспресс»

Приложение «Аптека Вита экспресс» является типичным примером мобильного приложения для аптеки. Интерфейс приложения приведен на рисунке 1.

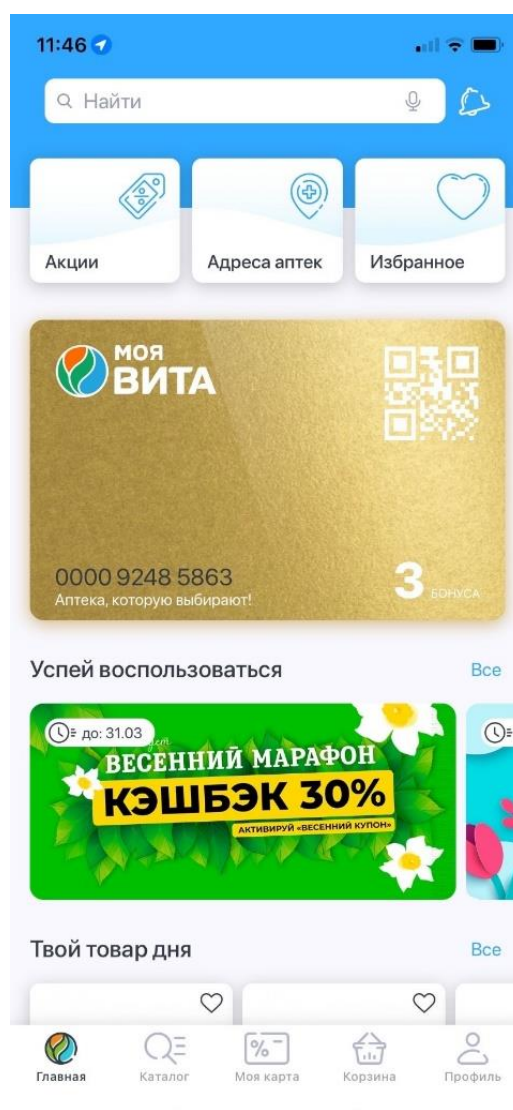


Рисунок 1 – Интерфейс приложения «Аптека Вита»

Среди достоинств данного приложения можно выделить то, что оно предоставляет пользователю возможность поиска лекарств в каталоге товаров, просмотра актуальных цен, отображает адреса всех торговых точек сети, а также дает возможность авторизоваться и участвовать в программе лояльности [1].

Главным недостатком, с точки зрения пользователя, можно считать невозможность сравнения цен с аналогичными товарами в других аптеках города.

1.2.2 Мобильное приложение «Аптека.ру»

Федеральная сеть по заказу лекарств «Аптека.ру» работает напрямую с поставщиками фармацевтических товаров, поэтому цены на многие лекарства

ниже средней по России [2]. Данный сервис имеет мобильное приложение, интерфейс которого представлен на рисунке 2.



Рисунок 2 – Интерфейс приложения «Аптека.ру»

Среди достоинств этого приложения можно выделить то, что оно предоставляет пользователю возможность легко и быстро найти необходимое лекарство, а также товары-аналоги. Созданный заказ оплачивается вне зависимости от стоимости лекарств в выбранной аптеке, цена на товары определена сервисом. Кроме того, существует программа лояльности.

Недостатком этого приложения является отсутствие возможности сравнения стоимости лекарств в разных торговых сетях города.

1.2.3 Мобильное приложение «Еаптека»

Сервис «ЕАПТЕКА» является одной из крупнейших в России интернет-аптек [3]. Недавно у него появилось мобильное приложение, интерфейс которого представлен на рисунке 3.

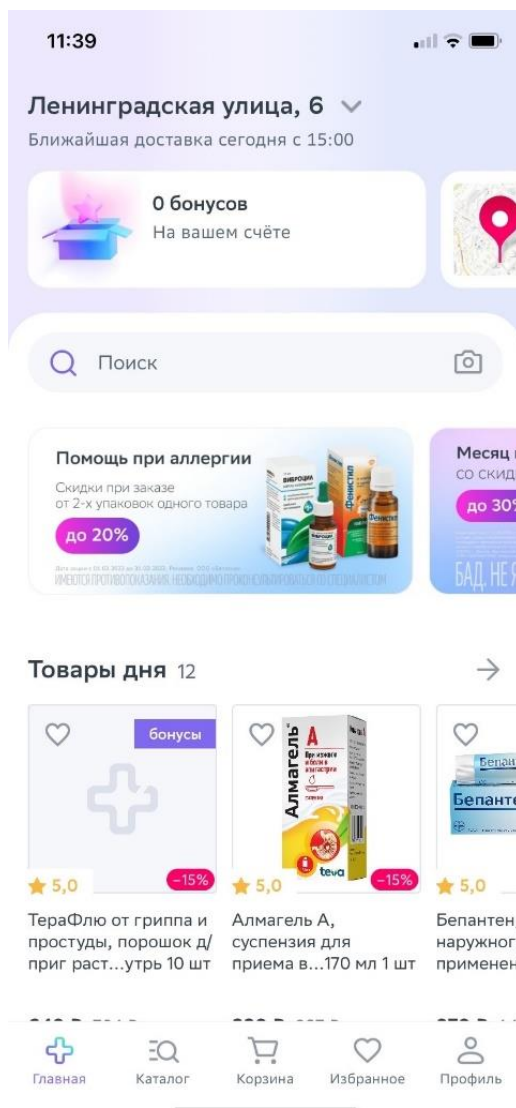


Рисунок 3 – Интерфейс приложения «Еаптека»

Главным достоинством приложения является то, что оно позволяет сравнить цены на лекарства в разных торговых сетях города Пенза. Кроме того, поиск лекарств и их аналогов в каталоге товаров осуществляется просто и быстро.

Основными недостатками является то, что сравнить стоимость товара можно лишь уже на этапе оформления заказа.

1.2.4 Мобильное приложение «Все аптеки»

Сервис «Все аптеки» – агрегатор товаров из аптечных сетей. Интерфейс мобильного приложения «Все аптеки» представлен на рисунке 4.

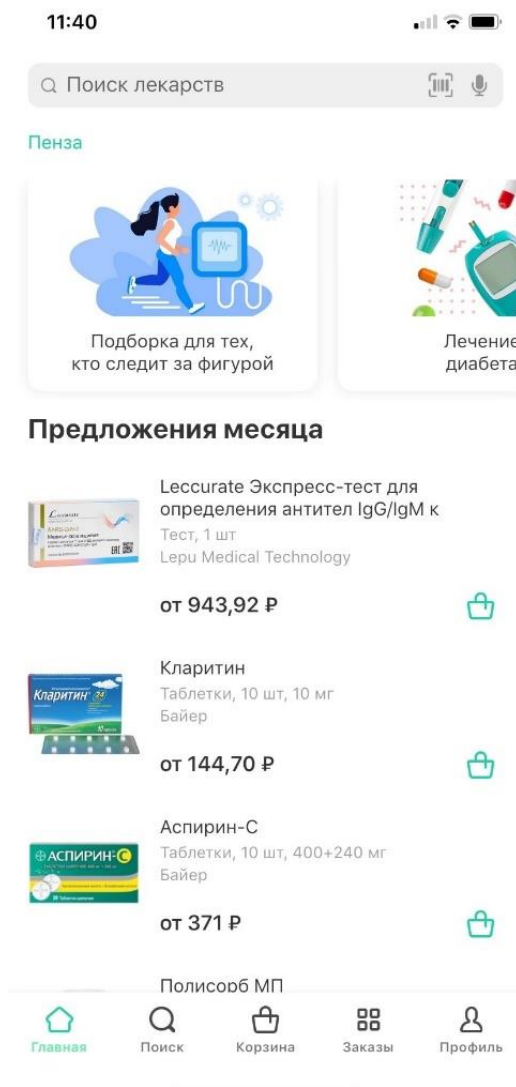


Рисунок 4 – Интерфейс приложения «Все аптеки»

Ресурс позволяет найти лекарство или его аналоги в каталоге товаров и дает возможность сравнить цены на одни и те же товары в разных аптеках города [4].

К недостаткам можно отнести возможность увидеть актуальные цены в разных аптечных сетях города только на этапе оформления заказа.

1.2.5 Мобильное приложение «Здравсити»

Цифровой сервис «Здравсити» работает более чем в 80 регионах России [5]. Как и многие другие интернет-сервисы, он имеет мобильное приложение, интерфейс которого приведен на рисунке 5.

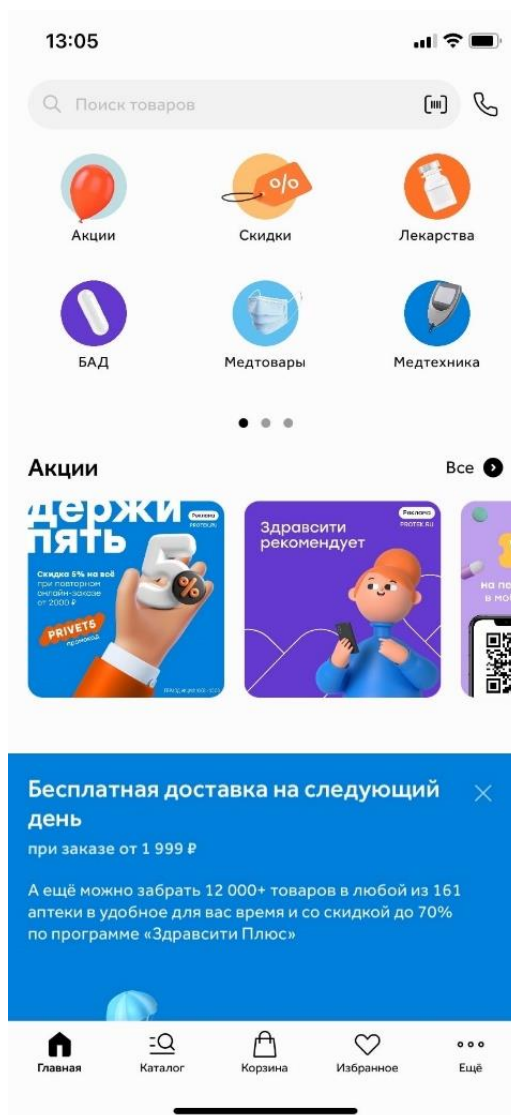


Рисунок 5 – Интерфейс приложения «Здравсити»

Данное приложение обеспечивает возможность легко и быстро найти лекарство в каталоге товаров и сравнить его с аналогичными товарами, имеющими точно такое же действующее вещество. Однако, отсутствует возможность сравнения цен на одинаковые лекарства в разных аптеках. Существенным минусом также является то, что в городе Пенза к данному сервису в настоящее время подключена только одна аптека.

1.2.6 Мобильное приложение «Мегааптека»

Сервис «Мегааптека.ру» – это онлайн-агрегатор аптек, к которому в настоящий момент к сервису подключено более 100 аптечных сетей по всей России [6]. Интерфейс приложения «Мегааптека» представлен на рисунке 6.



Рисунок 6 – Интерфейс приложения «Мегааптека»

Приложение предоставляет пользователю возможность найти товар и его аналоги в каталоге товаров. Достоинством является возможность сравнить цены на лекарства в аптеках разных торговых сетей города Пенза. Но значительный недостаток в том, что это сравнение доступно лишь на этапе оформления заказа.

1.2.7 Сравнение мобильных приложений-агрегаторов аптечных товаров

Проведем сравнительный анализ рассмотренных выше программных средств.

Для оценки приложений будет использоваться трехбалльная шкала оценки, в которой 0 означает «полное отсутствие реализации», 1 – «частичную реализацию», а 2 – «полную реализацию» рассматриваемой функции.

Результаты сравнения мобильных приложений-агрегаторов аптечных товаров представлены в таблице 1.

Таблица 1 – Сравнение программных средств аналогичного назначения

Критерий \ Приложение	Вита экспресс	Аптека.ру	Еаптека	Все аптеки	Здравсити	МегаАптека
1. Возможность осуществить заказ в приложении	2	2	2	2	2	2
2. Возможность просмотра информации о лекарственных препаратах	2	2	2	2	2	2
3. Возможность поиска лекарства в системе	2	2	2	2	2	2
4. Возможность просмотра истории заказов	2	2	2	1	2	2
5. Сравнение цен в разных аптечных сетях города Пенза	0	0	1	1	0	1
6. Реализация подключения различных аптечных сетей к сервису	0	2	2	2	1	2
7. Возможность зарегистрироваться и авторизоваться	2	2	2	2	2	2
8. Наличие программы скидок, акций и персональных приложений и подборок на главной странице	2	2	2	2	2	2
9. Возможность изменения цвета темы приложения	0	2	0	0	0	0
10. Наличие программы лояльности	2	2	2	0	2	0
ИТОГО	14	18	17	14	15	15

Для наглядности, полученные результаты отображены на лепестковой диаграмме, приведенной на рисунке 7.



Рисунок 7 – Результаты сравнения приложений-аналогов

Подводя итог сравнения можно сделать вывод, что ни одно из рассмотренных приложений не соответствует объявленным критериям в полной мере. Среди всех продуктов-аналогов большее количество баллов набрал сервис «Аптека.ру», однако он не обеспечивает пользователям возможность сравнения цен в различных аптечных сетях.

В результате принято решение разработать собственное программное средство, которое будет включать в себя лучшие решения от рассмотренных приложений и будет лишено наиболее существенных недостатков.

Исходя из данной таблицы, к разрабатываемому приложению были выдвинуты следующие дополнительные требования:

- приложение должно выводить на экран пользователя актуальные цены на выбранный товар в разных аптеках города Пенза;
- приложение должно обеспечивать пользователю возможность осуществления заказа;
- приложение должно обеспечивать пользователю возможность выбора темы интерфейса;
- приложение должно предоставлять пользователям доступ к программе лояльности.

1.3 Анализ функциональных требований

Обобщив сказанное выше, к разрабатываемому мобильному приложению были выделены следующие функциональные требования:

- обеспечение возможности регистрации
- обеспечение возможности авторизации;
- просмотр списка лекарственных препаратов;
- просмотр подробных данных о выбранном препарате;
- визуализация информации о ценах в различных аптечных сетях;
- поиск в системе лекарства по его названию;
- возможность осуществить заказ;
- возможность изменить цвет темы интерфейса;
- возможность просмотра истории заказов.

В соответствии с выявленными требованиями была разработана диаграмма вариантов использования UML, представленная на рисунке 8 [7].

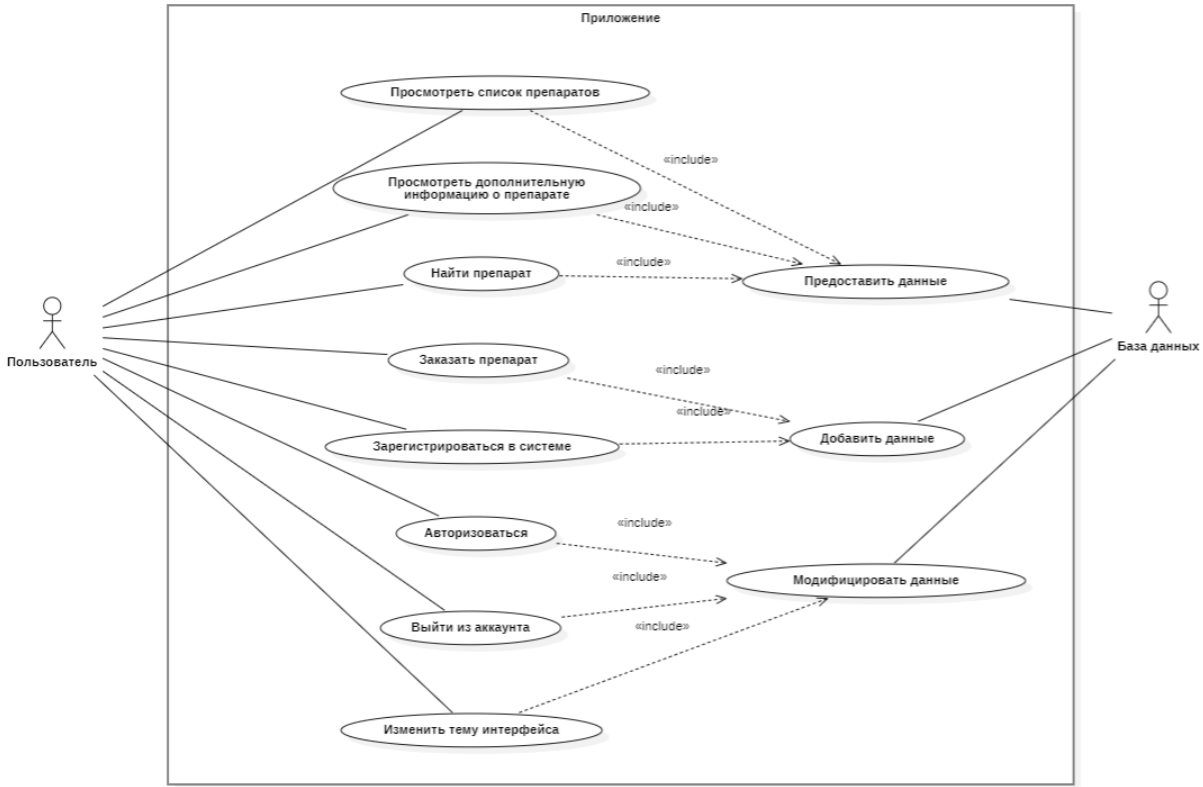


Рисунок 8 – Диаграмма вариантов использования

Сценарий варианта использования «Посмотреть дополнительную информацию о препарате» представлен на рисунке 9.

Прецедент: Посмотреть дополнительную информацию о препарате
ID: 1
Краткое описание: Пользователь просматривает стоимость лекарств
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь открыл приложение; 2. Пользователь открыл список лекарственных средств.
Основной поток: 1. Пользователь нашел интересующий его препарат. 2. Пользователь произвел нажатие на экран в области отображения элемента.
Постусловия: 1. Приложение отображает на экране смартфона стоимость выбранного лекарства в различных аптечных сетях города
Альтернативные потоки: Нет

Рисунок 9 – Сценарий варианта использования «Посмотреть дополнительную информацию о препарате»

Сценарий варианта использования «Зарегистрироваться» изображен на рисунке 10.

Прецедент: Зарегистрироваться
ID: 2
Краткое описание: Пользователь регистрируется в приложении
Главные актеры: Пользователь, приложение
Второстепенные актеры: Нет
Предусловие: Нет
Основной поток: 1. Если пользователь вводит некорректные данные (пароль короче 6 символов) 1.1. Приложение выдает предупреждение 2. Иначе, если пользователь с указанным номером телефона уже существует 2.2. Приложение выдает предупреждение 3. Иначе 3.1. Пользователь добавляется в базу данных
Постусловия: 1. Новый пользователь сохранен в базе данных приложения
Альтернативные потоки: Нет

Рисунок 10 – Сценарий варианта использования «Зарегистрироваться»

Сценарий варианта использования «Посмотреть дополнительную информацию о препарате» представлен на рисунке 11.

Прецедент: Изменить тему интерфейса
ID: 3
Краткое описание: Пользователь изменяет тему интерфейса приложения
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь открыл приложение; 2. Пользователь авторизован; 3. Пользователь открыл вкладку «Профиль»
Основной поток: 1. Пользователь нажал переключатель «Темная тема».
Постусловия: 1. Тема приложения изменена
Альтернативные потоки: Нет

Рисунок 11 – Сценарий варианта использования «Посмотреть дополнительную информацию о препарате»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования были представлены их спецификации, которые позволяют отобразить последовательность действий пользователя и системы.

1.4 Анализ нефункциональных требований

Разработка базы данных будет проводиться с использованием Microsoft SQL Server 18 на языке T-SQL. База данных будет размещена на удаленном сервере.

Клиентская часть мобильного приложения будет написана на языке C# с использованием фреймворка Xamarin в интегрированной среде разработки Microsoft Visual Studio 2022 [8].

Xamarin представляет собой платформу с открытым исходным кодом. Она предназначена для построения графического интерфейса производительных приложений для различных операционных систем с .NET. Платформа Xamarin представляет собой уровень абстракции, обеспечивающий возможность управления взаимодействием между общим кодом и кодом базовой платформы. Благодаря Xamarin основная часть кода приложений может использоваться на разных платформах без изменений [9].

Клиентское приложение должно быть отлажено и успешно работать под управлением операционной системы Android версии 5.0 и более новой.

2 Проектирование мобильного приложения-агрегатора аптечных товаров

2.1 Архитектура приложения

В разрабатываемом мобильном приложении было принято использовать архитектуру «клиент-сервер», схема которой приведена на рисунке 12.



Рисунок 12 – Архитектура «клиент-сервер»

На сервере будет расположена база данных.

Клиентское приложение будет производить обращения к серверу и предоставлять пользователю удобный интерфейс для просмотра и взаимодействия с данными из базы [10].

2.2 Проектирование структур данных

Для эффективной работы приложения требуется следующая информация о конкретном лекарственном препарате:

- категория;
- название;
- изображение;
- действующее вещество;
- форма выпуска;
- стоимость;
- аптечная сеть, осуществляющая реализацию.

Также требуется информация о пунктах выдачи заказа, а именно адреса аптек, входящих в каждую торговую сеть.

2.2.1 Концептуальная модель базы данных

На основе анализа предметной области и результатов моделирования бизнес-процессов была составлена концептуальная модель базы данных в виде диаграммы «сущность-связь», представленной на рисунке 13 [11].

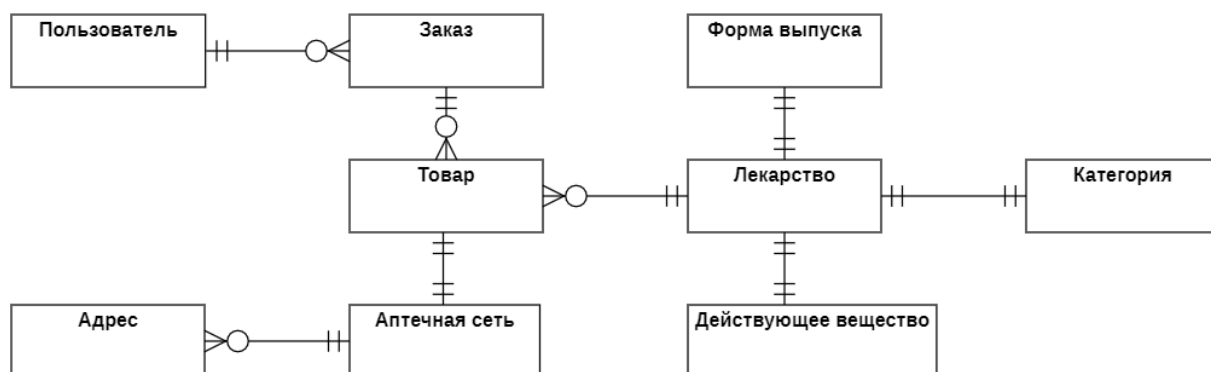


Рисунок 13 – Концептуальная модель базы данных предметной области
«Аптека»

На основе концептуальной модели базы данных была составлена таблица сущностей, приведенная в таблице 2.

Таблица 2 – Сущности БД

Название сущности	Идентификатор сущности	Назначение сущности
Пользователь	User	Содержит сведения о пользователях приложения
Заказ	Order	Содержит сведения о заказах, совершенных пользователями
Товар	ConcretMed	Содержит сведения о стоимости лекарств в различных аптеках
Лекарство	Med	Содержит сведения о лекарствах
Форма выпуска	Form	Содержит сведения о форме выпуска препарата
Категория	Category	Содержит сведения о категории, к которой относится лекарство
Действующее вещество	Active	Содержит сведения о действующем веществе в составе препарата
Аптечная сеть	Pharma	Содержит сведения о названиях аптечных сетей
Адрес	Place	Содержит сведения о адресах аптек, входящих в аптечную сеть

Наборы атрибутов для каждой сущности предметной области и их характеристики приведены в приложении А.

2.2.2 Логическая модель базы данных

Логическая модель – это расширение концептуальной модели данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила [12].

На основании концептуальной модели была сконструирована логическая модель базы данных изображенная на рисунке Б.1 в приложении Б.

Логическая модель построена в среде Visual Paradogm 17.0 в нотации IDEF1X.

2.3 Проектирование пользовательского интерфейса

На основе выявленных функциональных требований был разработан интерфейс мобильного приложения [13]. Он включает в себя страницы входа, регистрации, список имеющихся в системе категорий, лекарств и результатов поиска, подробной информации о препарате и информации о пользователе.

Макет страницы входа представлен на рисунке 14. На нем имеются следующие элементы:

- 1 – поле ввода номера мобильного телефона (логина) пользователя;
- 2 – поле ввода пароля пользователя;
- 3 – кнопка «Вход»;
- 4 – кнопка перехода на страницу регистрации.

При нажатии на кнопку «Вход» система проверяет, существует ли пользователь с таким логином и паролем. Если такого пользователя нет, будет выведено сообщение о неверном логине или пароле. В противном случае авторизация будет считаться выполненной успешно.

The diagram shows a vertical rectangular frame containing four elements stacked vertically. To the right of each element is a horizontal line pointing to a number. The elements are: 1. An empty rectangular input field. 2. Another empty rectangular input field. 3. A rectangular button with the text 'Войти' (Login). 4. A rectangular button with the text 'Регистрация' (Registration), preceded by the text 'Еще не зарегистрировались?' (Not yet registered?).

Рисунок 14 – Макет страницы входа

Макет страницы регистрации представлен на рисунке 15. На нем имеются следующие элементы:

- 1 – поле ввода номера мобильного телефона;
- 2 – поле ввода пароля пользователя;
- 3 – кнопка «Регистрация»;
- 4 – кнопка перехода на страницу входа.

При нажатии кнопки «Регистрация» система проверяет, являются ли введенные логин и пароль допустимыми. Если нет, будет выведено сообщение. Иначе регистрация будет считаться успешной и будет проведена автоматическая авторизация.



Рисунок 15 – Макет страницы регистрации

Макет страниц категорий, списка товаров и результатов поиска приведен на рисунке 16. На нем имеются следующие общие элементы:

- 1 – поле ввода текста для поиска;
- 2 – области экрана, где будет отображен результат;
- 3 – панель навигации.

В качестве результата будет отображено изображение и название фармацевтического товара, при нажатии на которые отобразится страница с подробной информацией о лекарственном препарате.

Панель навигации позволит перемещаться между главной страницей, страницей поиска и страницей пользователя.

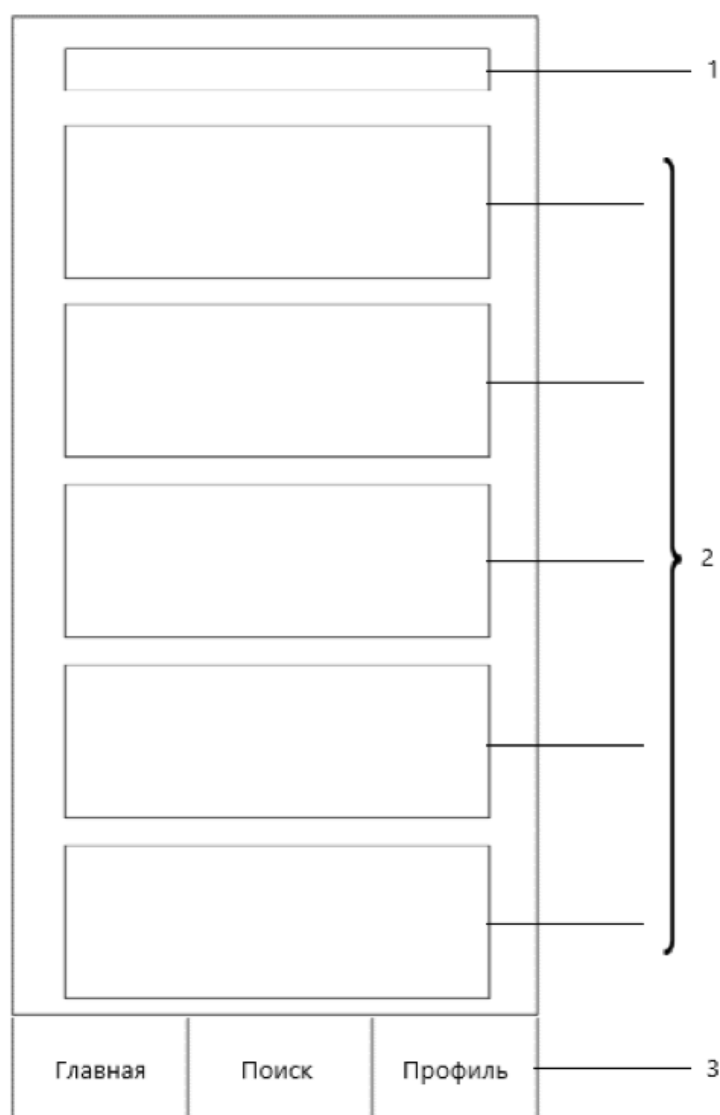


Рисунок 16 – Макет страницы вывода списковой информации

Макет страницы пользователя приведен на рисунке 17. На нем имеются следующие элементы:

- 1 – поле для отображения логина пользователя;
- 2 – поле для отображения города;
- 3 – переключатель цвета темы интерфейса;
- 4 – кнопка истории заказов;
- 5 – кнопка для выхода из аккаунта;
- 6 – навигационная панель.

При взаимодействии с переключателем тема интерфейса приложения будет изменена на противоположную.

При нажатии на кнопку «История заказов» будет отображена страница, содержащая информацию о совершенных пользователем заказах.

При нажатии на кнопку «Выход» будет произведен выход из аккаунта пользователя и отобразится страница авторизации.

The diagram illustrates the layout of a user profile page. It consists of a main content area and a bottom navigation bar. The main content area contains the following elements from top to bottom:

- A label "Логин:" followed by a text input field (callout 1).
- A label "Город:" followed by a text input field (callout 2).
- A label "Темная тема" followed by a toggle switch (callout 3).
- A button labeled "История заказов" (callout 4).
- A button labeled "Выход" (callout 5).

The bottom navigation bar (callout 6) contains three buttons: "Главная", "Поиск", and "Профиль".

Рисунок 17 – Макет страницы пользователя

Макет страницы подробной информации о препарате приведен на рисунке 18. На рисунке имеются следующие элементы:

- 1 – область для отображения названия препарата;
- 2 – область для отображения изображения;
- 3 – область вывода стоимости лекарственного средства в разных аптеках;
- 4 – навигационная панель.

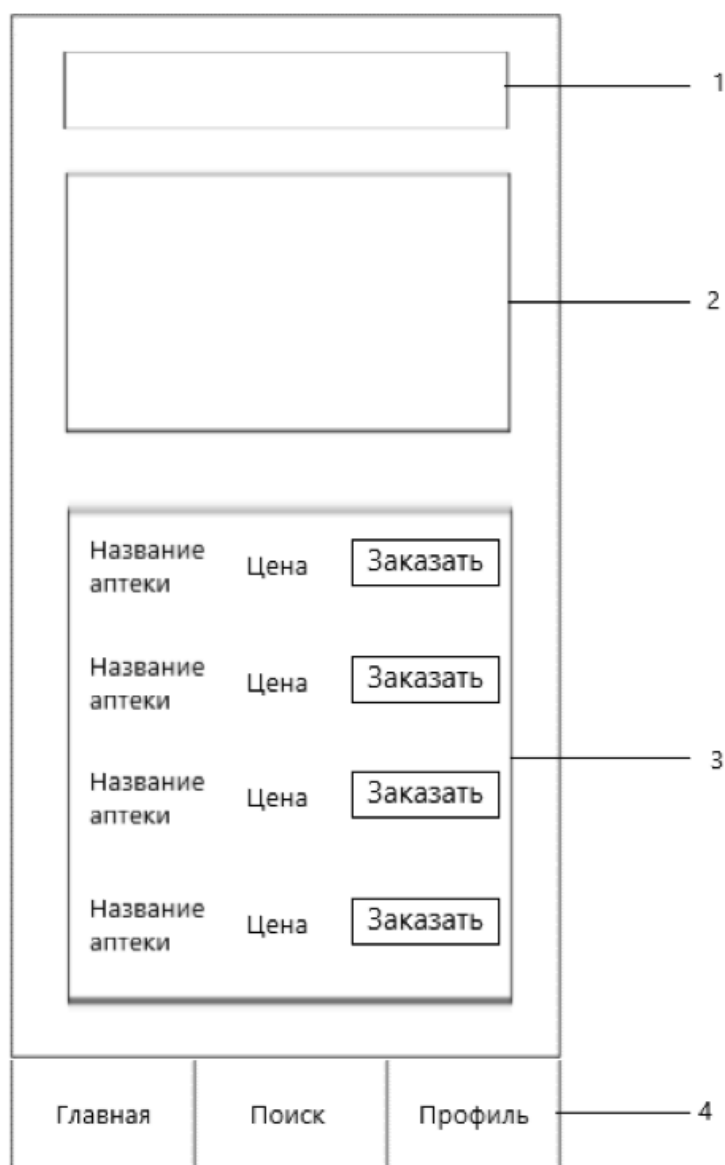


Рисунок 18 – Страница подробной информации о препарате

При выводе стоимости препарата на экране будут отображены наименования аптечных сетей, в которых осуществляется продажа выбранного лекарственного средства, а также его стоимость в рублях и кнопка для совершения заказа.

2.4 Проектирование программных средств

На основе логической модели базы данных была спроектирована диаграмма классов мобильного приложения, представленная на рисунке Б.2 в приложении Б.

Краткое описание функционального назначения представленных на диаграмме классов приведено в таблице 3.

Таблица 3 – Компоненты диаграммы классов

Компонент	Назначение
App	Точка входа в программу
AppShell	Класс, реализующий интерфейс навигационного меню
MainPage	Класс, реализующий интерфейс главной страницы приложения
MainViewModel	Класс для работы с данными главной страницы
CategoryPage	Класс, реализующий интерфейс страницы категорий лекарственных препаратов
CategoryViewModel	Класс для работы с данными о категориях препаратов
Category	Класс, содержащий информацию о категории, к которой относится лекарственный препарат
ItemsPage	Класс, реализующий интерфейс страницы списка лекарственных препаратов
ItemsViewModel	Класс для работы с данными о лекарственных препаратах
Item	Класс, содержащий информацию о лекарственном препарате
ItemsDetailPage	Класс, реализующий интерфейс отображения информации о выбранном препарате
ItemsDetailViewModel	Класс для работы с данными выбранного лекарственного препарата
Pharm	Класс, содержащий информацию о цене лекарства в аптечной сети
OrderPage	Класс, реализующий интерфейс страницы заказа
OrderViewModel	Класс для работы с данными заказа
Order	Класс, содержащий информацию о заказе
NotYetPage	Класс, реализующий интерфейс пустой страницы
NotYetViewModel	Класс для работы с данными пустой страницы
ProfilePage	Класс, реализующий интерфейс страницы профиля пользователя
ProfileViewModel	Класс для работы с данными профиля
User	Класс, содержащий информацию о пользователе

В ходе разработки приложения будет реализовано 22 класса, 9 из которых отвечаю за визуальную функциональность приложения, другие за работу с данными предметной области.

3 Реализация мобильного приложения-агрегатора аптечных товаров

3.1 Реализация базы данных

На основе построенной ранее логической модели базы данных при помощи системы Visual Paradigm 17.0 была построена ее физическая модель, представленная на рисунке 19.

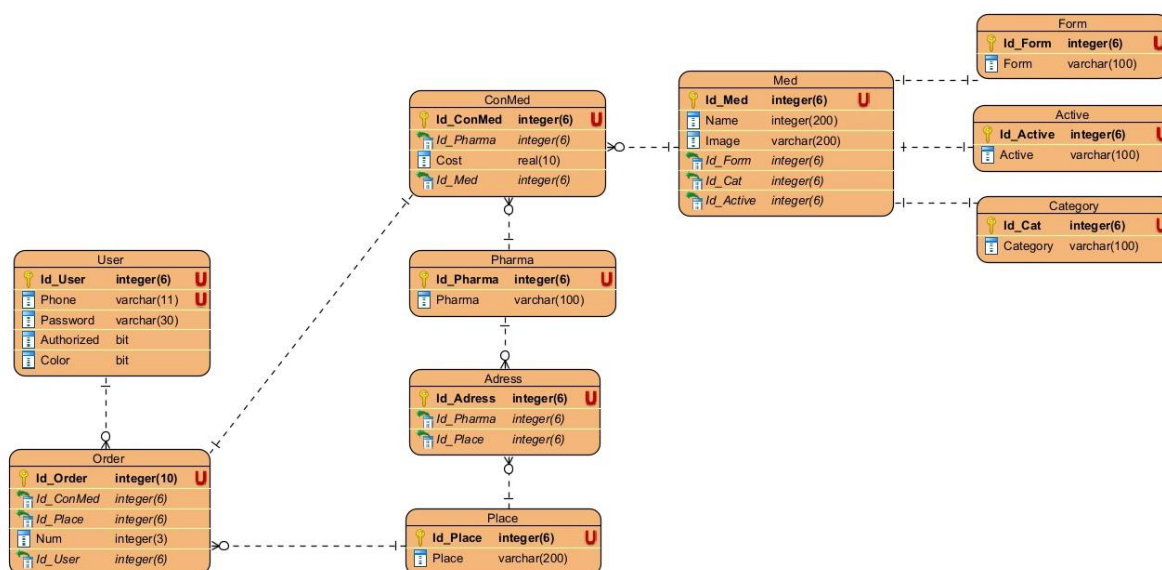


Рисунок 19 – Физическая модель базы данных

На основе составленной физической модели БД был сгенерирован скрипт на языке SQL для создания базы данных. Его код представлен в приложении В. Развертывание спроектированной базы данных было выполнено путем запуска данного скрипта под управлением СУБД Microsoft SQL Server 2018.

3.2 Наполнение базы тестовыми данными

Созданная база данных содержит 10 сущностей. Данные, внесенные в основные таблицы представлены на рисунках 20 – 24.

Содержимое таблицы аптечных сетей приведено на рисунке 20.

	id_pharm	namePh
1	1	Аптека Вита Экспресс
2	2	Аптека Фармация
3	3	Аптека Апрель
4	4	Магнит аптека
5	5	Аптеки Плюс

Рисунок 20 – Данные в таблице «Pharma»

Содержимое таблицы адресов аптек приведено на рисунке 21.

	id_adr	adress
1	1	г. Пенза, ул. Мира, д. 6
2	2	г. Пенза, ул. Московская, д. 83
3	3	г. Пенза, ул. 8 Марта, д. 27 А
4	4	г. Пенза, ул. Кижеватова, д. 15 А
5	5	г. Пенза, ул. Коммунистическая, д. 36
6	6	г. Пенза, ул. Ленина, д. 19
7	7	г. Пенза, ул. Карпинского, д. 37
8	8	г. Пенза, ул. Пушкина, д. 15
9	9	г. Пенза, ул. Антонова, д. 15
10	10	г. Пенза, ул. Попова, д. 22

Рисунок 21 – Данные в таблице «Adress»

Содержимое таблицы цен на лекарства приведено на рисунке 22.

	id_conMed	id_pharm	cost	id_med
1	1	3	110	1
2	2	1	113	1
3	3	5	34,8	2
4	4	2	35,6	2
5	5	4	37,9	2
6	6	3	20	3
7	7	4	21,5	3
8	8	5	17,2	3
9	9	3	96	4
10	10	1	92	4
11	11	2	87	4
12	12	3	553	5
13	13	4	429	5
14	14	5	326	5
15	15	3	602	6
16	16	4	506,5	6
17	17	5	369,3	6

Рисунок 22 – Данные в таблице «ConMed»

Содержимое таблицы категорий приведено на рисунке 23.

	id_cat	category
1	1	Обезболивающее
2	2	Жаропонижающее
3	3	От боли в горле
4	4	От насморка
5	5	От аллергии
6	6	Противовоспалительное

Рисунок 23 – Данные в таблице «Category»

Содержимое таблицы лекарств приведено на рисунке 24.

	id_med	id_activ	id_form	img	nameMed	id_cat
1	1	1	1	https://irecommend.ru/sites/default/files/imagecach...	Анальгин Реневал 20 таб.	1
2	2	1	1	https://apteki.meds.ru/thumb/400x400x1N/upload/ib...	Анальгин Татхимфарм 20 таб.	1
3	3	2	1	https://pictures1.apteka-april.ru/products/191611/80...	Ацетилсалициловая кислота Медисорб 20 таб.	6
4	4	2	1	https://pictures1.apteka-april.ru/products/70618/800...	Аспирин кардио 28 таб.	6
5	5	3	1	https://grammidin.ru/local/templates/new_grammid...	Граммидин Нео с анестетиком 18 таб.	3
6	6	3	3	https://grammidin.ru/local/templates/new_grammid...	Граммидин с анестетиком, спрей 112 доз	3
7	7	4	1	https://images.apteka.ru/original_23ff4e73-4ebb-49...	Лоратадин OZON 10 таб.	5
8	8	4	1	https://f.stolichki.ru/s/drugs/large/68/68585.jpg	Лоратадин Реневал 30 таб.	5
9	9	5	1	https://f.stolichki.ru/s/drugs/large/10/10683.jpg	Кагоцел 12мг 10 таб.	2
10	10	6	1	https://f.stolichki.ru/s/drugs/large/45/45054.jpg	Дротаверин Реневал 56 таб.	1
11	11	6	1	https://f.stolichki.ru/s/drugs/large/41/41183.jpg	Но-шпа 40мг 64 таб.	1
12	12	7	1	https://f.stolichki.ru/s/drugs/large/63/63087.jpg	Парацетамол Реневал 10 таб.	2
13	13	7	1	https://f.stolichki.ru/s/drugs/large/48/48083.jpg	Парацетамол Озон 20 таб.	2
14	14	8	2	https://f.stolichki.ru/s/drugs/large/59/59078.jpg	Снуп спрей назальный 15мл	4
15	15	8	2	https://f.stolichki.ru/s/drugs/large/44/44935.jpg	Ксилен спрей назальный 15мл	4

Рисунок 24 – Данные в таблице «Med»

Эти данные используются для формирования представлений и последующего вывода на экран пользователя.

3.3 Реализация хранимых процедур и функций

Для взаимодействия мобильного приложения с информацией из таблиц базы данных было создано несколько хранимых процедур:

- FindUser, которая проверяет, есть ли в базе пользователь с указанным логином;
- GetCats, которая выводит все имеющиеся в базе категории лекарственных препаратов;
- GetMedsFromCat, которая выводит все лекарственные препараты, относящиеся к указанной категории;

– GetMedInfo, которая выводит информацию о ценах на выбранное лекарство в разных аптеках;

– FindApt, которая выводит все адреса указанной аптечной сети.

Листинги данных процедур на языке T-SQL приведены в приложении Г.

3.4 Реализация клиентской части приложения

На рисунке Б.3 в приложении Б представлена диаграмма компонентов разработанного мобильного приложения.

Описание основных компонентов мобильного приложения приведено в таблице 4.

Таблица 4 – Компоненты приложения

Компонент	Назначение
App.xaml	Определяет ресурсы, общие для всего приложения
App.xaml.cs	Точка входа в программу
AppShell.xaml	Реализует интерфейс навигационного меню
AppShell.xaml.cs	Реализует взаимодействие с навигационным меню
LoginPage.xaml	Реализует интерфейс страницы авторизации
LoginPage.xaml.cs	Реализует взаимодействие с элементами интерфейса страницы авторизации
LoginViewModel.cs	Реализует бизнес логику процесса авторизации
RegPage.xaml	Реализует интерфейс страницы регистрации
RegPage.xaml.cs	Реализует взаимодействие с элементами интерфейса страницы регистрации
RegViewModel.cs	Реализует бизнес логику процесса регистрации
MainPage.xaml	Реализует интерфейс главной страницы
MainPage.xaml.cs	Реализует взаимодействие с элементами интерфейса главной страницы
MainViewModel.cs	Реализует бизнес логику главной страницы
CategoryPage.xaml	Реализует интерфейс страницы категорий
CategoryPage.xaml.cs	Реализует взаимодействие с элементами интерфейса страницы категорий
CategoryViewModel.cs	Реализует бизнес логику страницы категорий

Продолжение таблицы 4

Category.cs	Класс, содержащий информацию о категории, к которой относится лекарственный препарат
ItemsPage.xaml	Реализует интерфейс страницы списка товаров внутри категории
ItemsPage.xaml.cs	Реализует взаимодействие с элементами интерфейса страницы списка товаров внутри категории
ItemsViewModel.cs	Реализует бизнес логику страницы списка товаров внутри категории
ItemsSearchPage.xaml	Реализует интерфейс страницы списка товаров при поиске
ItemsSearchPage.xaml.cs	Реализует взаимодействие с элементами интерфейса страницы списка товаров при поиске
ItemsSearchViewModel.cs	Реализует бизнес логику страницы списка товаров при поиске
Item.cs	Класс, содержащий информацию о лекарственном препарате
ItemsDetailPage.xaml	Реализует интерфейс страницы конкретного лекарства
ItemsDetailPage.xaml.cs	Реализует взаимодействие с элементами интерфейса страницы конкретного лекарства
ItemsDetailViewModel.cs	Реализует бизнес логику страницы конкретного лекарства
Pharm.cs	Класс, содержащий информацию о цене лекарства в аптечной сети
OrderPage.xaml	Реализует интерфейс страницы заказа
OrderPage.xaml.cs	Реализует взаимодействие с элементами интерфейса страницы заказа
OrderViewModel.cs	Реализует бизнес логику страницы заказа
Order.cs	Класс, содержащий информацию о заказе
NotYetPage.xaml	Реализует интерфейс пустой страницы
NotYetPage.xaml.cs	Реализует взаимодействие с элементами интерфейса пустой страницы
NotYetViewModel.cs	Реализует бизнес логику пустой страницы
ProfilePage.xaml	Реализует интерфейс страницы пользователя
ProfilePage.xaml.cs	Реализует взаимодействие с элементами интерфейса страницы пользователя
ProfileViewModel.cs	Реализует бизнес логику страницы пользователя
User.cs	Класс, содержащий информацию о пользователе

Разработка мобильного приложения проводилась на языке C# в среде разработки Microsoft Visual Studio 2022 под управлением операционной системы Windows 10 с использованием фреймворка Xamarin. Код приложения приведен в приложении Д.

3.5 Реализация интерфейса приложения

В соответствии с макетами интерфейса, представленными в пункте 2.3, были реализованы страницы мобильного приложения – агрегатора аптечных товаров с использованием языка разметки XAML фреймворка Xamarin.Forms.

Интерфейс страницы авторизации пользователя приведен на рисунке 25.

Номер телефона

Пароль

ВОЙТИ

Еще не зарегистрированы?

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 25 – Интерфейс страницы авторизации

Интерфейс страницы регистрации пользователя приведен на рисунке 26.

Номер телефона

Пароль

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 26 – Интерфейс страницы авторизации

Интерфейс главной страницы приведен на рисунке 27.

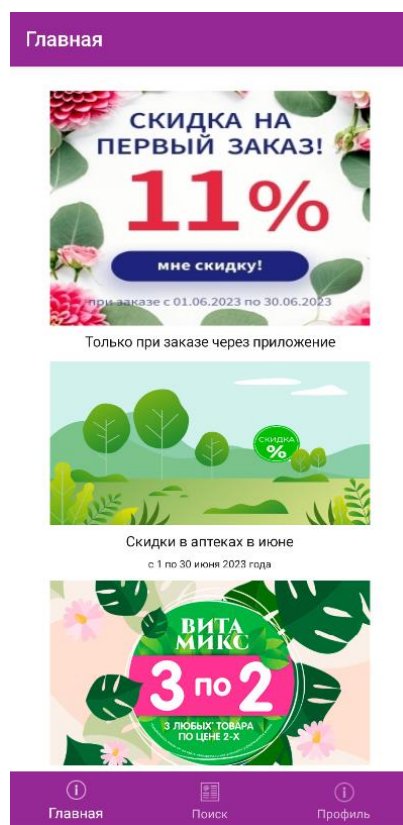


Рисунок 27 – Интерфейс главной страницы

Интерфейс страницы пользователя приведен на рисунке 28.

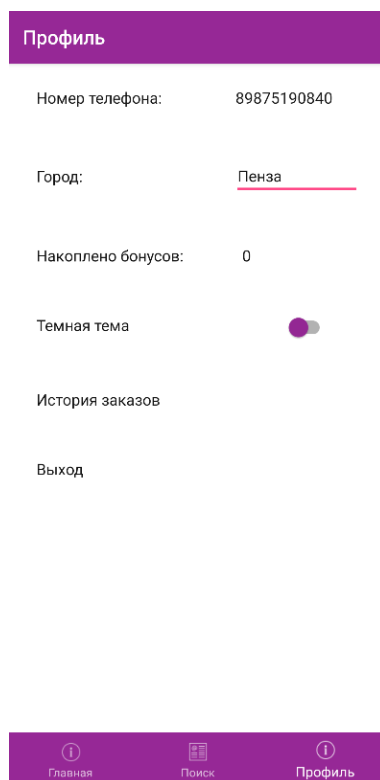


Рисунок 28 – Интерфейс страницы пользователя

Интерфейс страницы категорий товаров приведен на рисунке 29.

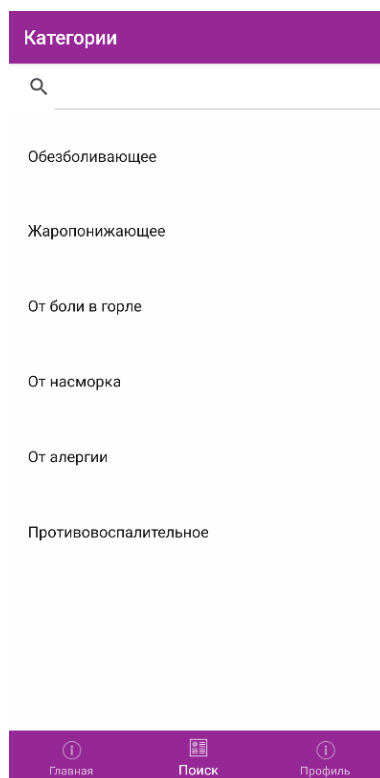


Рисунок 29 – Интерфейс страницы категорий товаров

Интерфейс страницы списка товаров приведен на рисунке 30.



Рисунок 30 – Интерфейс страницы списка товаров

Интерфейс страницы выбранного товара приведен на рисунке 31.

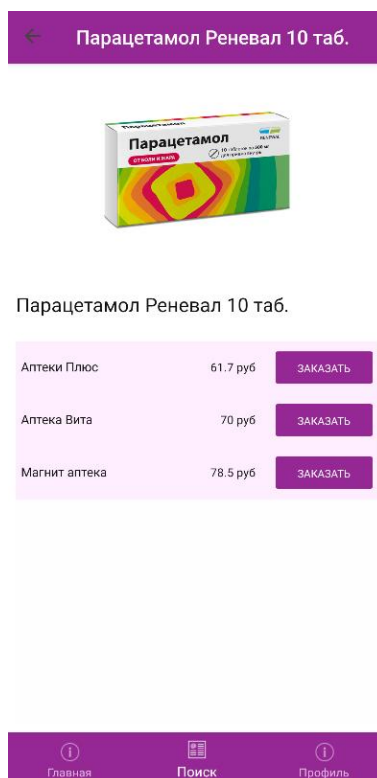


Рисунок 31 – Интерфейс страницы выбранного товара

Интерфейс страницы выбора адреса пункта выдачи приведен на рисунке 32.

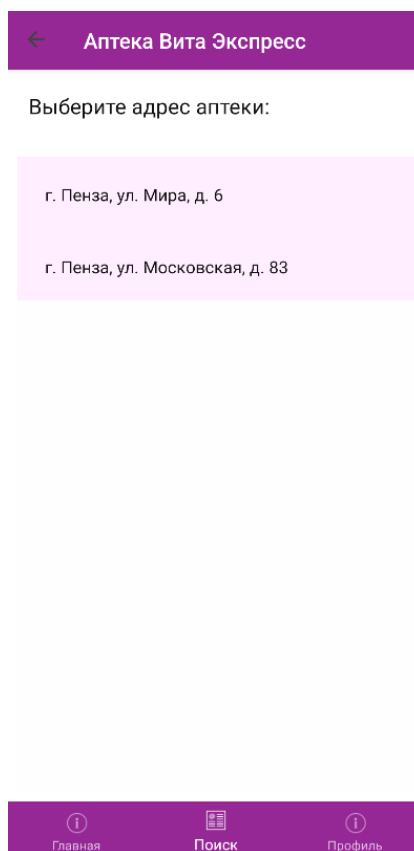


Рисунок 32 – Интерфейс страницы выбора адреса

С помощью Xamarin.Forms можно создавать приложения для операционных систем Android, iOS и Windows на основе общей базы кода. Xamarin.Forms позволяет создавать графический интерфейс на языке разметки XAML с помощью кода программной части в C#. Такие интерфейсы подготавливаются к просмотру на каждой платформе как собственные элементы управления [9].

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Существует достаточно много видов тестирования программных продуктов [14]. Наиболее популярны следующие виды тестирования:

- функциональное тестирование;
- модульное тестирование;
- интеграционное тестирование.

Функциональное тестирование уделяет основное внимание бизнес-требованиям к приложению. При этом проверяются только результат некоторого действия и не проверяются промежуточные состояния системы при выполнении этого действия [15].

Юнит тестирование или модульное тестирование работает на очень низком уровне, близко к исходному коду приложения. Способ заключается в тестировании отдельных методов и функций классов, компонентов или модулей, используемых в программном обеспечении. Модульные тесты могут выполняться очень быстро, потому что не требуют больших расходов на автоматизацию [15].

Интеграционное тестирование проверяет совместную работу используемых приложением модулей и сервисов. К этому виду тестирования относится проверка взаимодействия приложения с базой данных. Данный метод тестирования требует запуска различных компонентов приложения, поэтому является наиболее затратным [15].

При тестировании мобильных приложений необходимо учесть тот факт, что разные модели устройств имеют различные разрешения экранов, разные версии операционных систем и другие характеристики. Важно выявить, какие модели устройств наиболее популярны у пользователей, и проводить тестирование на этих моделях [16].

Необходимо также тестировать самую старую и самую новую операционные системы. Пользователь не сможет скачать приложение, если оно

не поддерживает установленную на его устройстве операционную систему. Разработчик может поставить предупреждение, что операционную систему следует обновить перед установкой приложения.

Следует учесть все возможные варианты взаимодействия пользователя с приложением. Для этого пишутся тест-кейсы с подробным описанием действий для проверки определенной функции. Тест-кейсы основаны на пользовательских сценариях – схемах того, как пользователь решает задачи с помощью приложения, и что ему может помешать.

Надо проверить взаимодействие пользователя с интерфейсом приложения, безопасность и удобство использования, производительность приложения и учесть все детали: от смены размера шрифта до параллельного использования с другими программами. Можно ли свернуть текущее приложение и зайти в другое и что ожидается от приложения при таких действиях [16].

Было принято решение осуществить проверку программы с помощью функционального тестирования. Такой выбор обусловлен тем, что к программе были предъявлены в основном функциональные требования, выполнение которых нужно проверить.

4.2 Функциональная диаграмма и таблица решений

Произведем функциональное тестирование процесса регистрации пользователя.

Пользователь должен иметь возможность авторизоваться при помощи номера мобильного телефона и пароля. Номер телефона должен содержать ровно 11 символов и допускается использовать только цифры. Пароль должен содержать от 6 до 30 символов. В пароле допускаются русские и латинские буквы, а также цифры. Невозможно зарегистрироваться под уже занятым логином.

Для того, чтобы построить эффективно-результативные тесты, можно воспользоваться методом функциональных диаграмм. При построении тестов с использованием данного метода в спецификации определяются причины и

следствия. Причина – входное условие, а следствие – выход системы. Далее спецификация преобразуется в управляющий граф.

Входные данные и выводы системы представлены в таблицах 5 и 6 соответственно.

Таблица 5 – Причины и следствия

П1	Корректная длина номера мобильного телефона
П2	Корректная длина пароля
П3	Номер мобильного телефона состоит из корректных символов
П4	Пароль состоит из корректных символов
П5	Номер мобильного телефона свободен

Таблица 6 – Следствия

С1	Сообщение: «Некорректная длина номера мобильного телефона»
С2	Сообщение: «Некорректная длина пароля»
С3	Сообщение: «Некорректные символы в составе номера мобильного телефона»
С4	Сообщение: «Некорректные символы в составе пароля»
С5	Сообщение: «Номер мобильного телефона занят»
С6	Сообщение: «Регистрация прошла успешно»

Для процесса регистрации пользователя была построена функциональная диаграмма, представленная на рисунке 33.

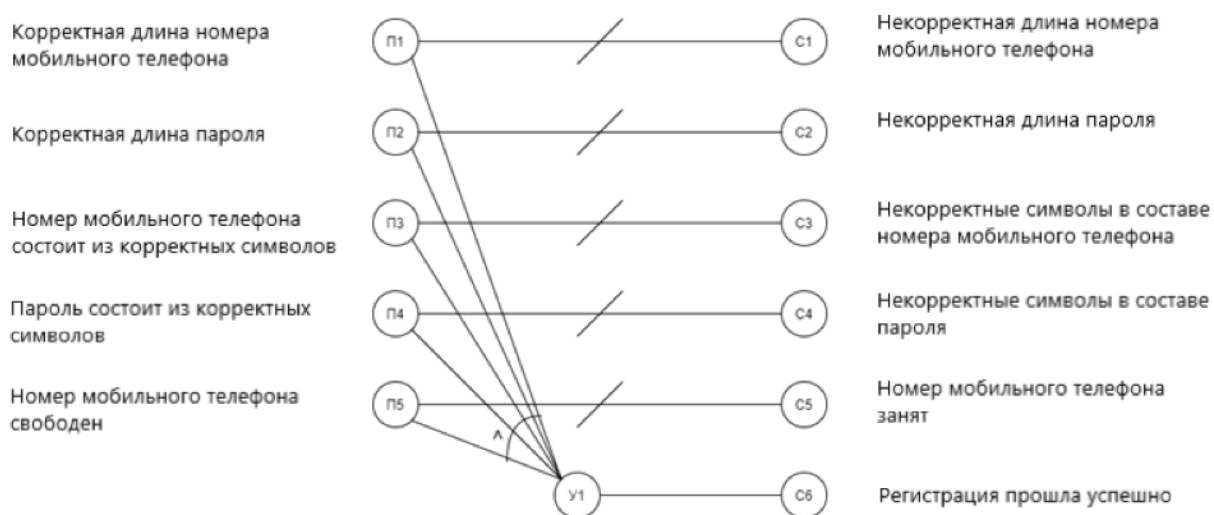


Рисунок 33 – Функциональная диаграмма

На основании функциональной диаграммы была построена таблица решений, представленная в таблице 7.

Таблица 7 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
П1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
П2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
П3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1
П4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
П5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
С1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0
С2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0
С3	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0
С4	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
С5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
С6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

В таблице 8 представлен набор тестов, составленные на основе таблицы решений.

Таблица 8 – Набор функциональных тестов

№	Входные данные	Ожидаемый результат	Результат
1	Логин: Ксения123 Пароль: psw-	Некорректная длина номера мобильного телефона; некорректная длина пароля; некорректные символы в составе номера мобильного телефона; некорректные символы в составе пароля	Тест пройден (рисунок Е.1)
2	Логин: Ксения12345 Пароль: 123-	Некорректная длина пароля; некорректные символы в составе номера мобильного телефона; некорректные символы в составе пароля	Тест пройден (рисунок Е.2)
3	Логин: Ксе Пароль: pass.word	Некорректная длина номера мобильного телефона; некорректные символы в составе номера мобильного телефона; некорректные символы в составе пароля	Тест пройден (рисунок Е.3)
4	Логин: ksenya89875 Пароль: pass.word	Некорректные символы в составе номера мобильного телефона; некорректные символы в составе пароля	Тест пройден (рисунок Е.4)
5	Логин: 8987519 Пароль: pwd-	Некорректная длина номера мобильного телефона; некорректная длина пароля; некорректные символы в составе пароля	Тест пройден (рисунок Е.5)
6	Логин: 89875190842 Пароль: pwd.1	Некорректная длина пароля; некорректные символы в составе пароля	Тест пройден (рисунок Е.6)

Продолжение таблицы 8

7	Логин: 8987519 Пароль: pass.word	Некорректная длина номера мобильного телефона; некорректные символы в составе пароля	Тест пройден (рисунок Е.7)
8	Логин: 89875190842 Пароль: pswd.1	Некорректные символы в составе пароля	Тест пройден (рисунок Е.8)
9	Логин: к89875 Пароль: pwd1	Некорректная длина номера мобильного телефона; некорректная длина пароля; некорректные символы в составе номера мобильного телефона	Тест пройден (рисунок Е.9)
10	Логин: к8987519084 Пароль: pwd1	Некорректная длина пароля; некорректные символы в составе номера мобильного телефона	Тест пройден (рисунок Е.10)
11	Логин: к8987 Пароль: password	Некорректная длина номера мобильного телефона; некорректные символы в составе номера мобильного телефона	Тест пройден (рисунок Е.11)
12	Логин: ksenya.k Пароль: password	Некорректные символы в составе номера мобильного телефона	Тест пройден (рисунок Е.12)
13	Логин: 89875 Пароль: pwd	Некорректная длина номера мобильного телефона; некорректная длина пароля	Тест пройден (рисунок Е.13)
14	Логин: 89875190842 Пароль: qwe12	Некорректная длина пароля	Тест пройден (рисунок Е.14)
15	Логин: 89875 Пароль: password	Некорректная длина номера мобильного телефона	Тест пройден (рисунок Е.15)
16	Логин: (занятый номер мобильного телефона) Пароль: password	Номер мобильного телефона занят	Тест пройден (рисунок Е.16)
17	Логин: (свободный номер мобильного телефона) Пароль: password	Регистрация прошла успешно	Тест пройден (рисунок Е.17)

Функциональное тестирование показало, что при регистрации пользователя приложение работает правильно.

4.3 Тестирование интерфейса приложения

При проведении юзабилити-тестирования работа приложения была проверена на смартфоне с операционной системой Android 8.0 и эмуляторе устройства Microsoft Visual Studio с операционной системой Android 13.0.

Для проверки корректности взаимодействия пользователя с интерфейсом приложения были приведены тесты, представленные в таблице 9.

Таблица 9 – Набор тестов интерфейса

№	Входные данные	Ожидаемый результат	Результат
1	Смена темы интерфейса приложения	Тема интерфейса изменена. Все элементы отображены корректно	Тест пройден (рисунок Е.18, а, рисунок Е.18, б)
2	Изменение ориентации экрана на альбомную	Ориентация экрана изменена. Все элементы отображены корректно	Тест пройден (рисунок Е.19, а, рисунок Е.19, б)

Проведенные тесты показали, что взаимодействие пользователя с интерфейсом приложения осуществляется корректно.

4.4 Метрики кода

4.4.1 Метрика SLOC

Количество строк кода (англ. Source Lines of Code – SLOC) – это метрика программного обеспечения, используемая для измерения его объёма с помощью подсчёта количества строк в тексте исходного кода [17].

Был проведен подсчет данной метрики в методе OnLoginClicked класса LoginViewModel, который проверяет наличие в базе данных пользователя с указанным номером телефона и паролем. Код функции приведен в листинге 1.

Листинг 1 – Метод OnLoginClicked

```
private async void OnLoginClicked(object obj)
{
    //Возврат, если есть предупреждение о неверном вводе
    if ((pwdlabel != "") || (loglabel != "")) return;
    try
    {
        //Строка соединения с БД
        SqlConnection cn = new SqlConnection("workstation
            id=KSENYAPHARMA.mssql.somee.com; packet size=4096;
            user id=kkk_SQLLogin_2; pwd=ydk4sfi7ym; data
            source=KSENYAPHARMA.mssql.somee.com; persist
            security info=False; initial catalog=KSENYAPHARMA");
        //Если соединение закрыто
        if(cn.State == System.Data.ConnectionState.Closed)
```

```

{
    //Открыть
    cn.Open();
    SqlCommand cmd = new SqlCommand("FindUser", cn);
    cmd.CommandType =
        System.Data.CommandType.StoredProcedure;
    //Добавить параметр
    SqlParameter l = new SqlParameter
    {
        ParameterName = "@1",
        Value = log
    };
    cmd.Parameters.Add(l);
    SqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        //Данные в БД найдены
        object id = reader.GetValue(0);
        object lo = reader.GetValue(1);
        object pw = reader.GetValue(2);
        object t = reader.GetValue(3);
        string th;
        if (Int32.Parse(t.ToString()) == 1)
            th = "DarkTheme";
        else
            th = "LightTheme";
        //Если пароль сходится с введенным
        if (pw.ToString() == pwd)
        {
            User u = new
                User(Int32.Parse(id.ToString()),
                    lo.ToString(), pw.ToString(), th, 1);
            //Получить пользователя
            selected = u;
        }
    }
    //Закреть соединение
    cn.Close();
}
}
catch (Exception ex)
{
    DisplayAlert("Error", ex.Message, "Ok");
}
//Если пользователь с введенными номером и паролем не найден
if (selected != null)
{
    await Shell.Current.GoToAsync($"://{nameof(AboutPage)}");
}
}
}

```

Физическими строками считаются все непустые строки текстового файла.

Измеряя логические строки кода, предпринимается попытка посчитать количество собственно операторов в программе, но, разумеется, их определение зависит от конкретного языка программирования.

В данной функции имеем:

- 57 физических строк кода;
- 47 логических строк кода;
- 10 строк комментариев.

Значение метрики SLOC равняется 57 физическим строкам кода и 47 логическим. Это нормальные значения для одной функции [17].

4.4.2 Цикломатическая сложность

При вычислении цикломатической сложности используется граф потока управления программы. Узлы графа соответствуют неделимым группам команд программы, они соединены ориентированными рёбрами, если группа команд, соответствующая второму узлу, может быть выполнена непосредственно после группы команд первого узла [17].

Граф потока управления программы для функции, представленной в листинге 1, приведен на рисунке 34.

Для вычисления цикломатической сложности $V(G)$ применяется следующая формула:

$$V(G) = e - n + 2p,$$

где e – количество дуг;

n – количество вершин;

p – число компонент связности.

Граф содержит 9 вершин, 13 дуг и один компонент связности.

$$V(G) = 13 - 9 + 2 = 6.$$

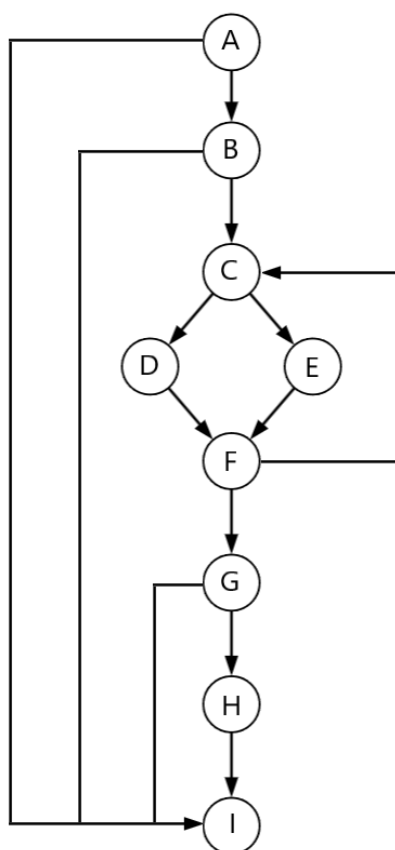


Рисунок 34 – Граф потока управления программы

Таким образом значение цикломатической сложности равно 6. Это значение не превышает 10, следовательно, функция не является структурно ложной [17].

4.4.3 Метрика Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы [18].

Основу метрики Холстеда составляют четыре измеряемые характеристики программы [19]:

- число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
- число уникальных операндов программы (словарь операндов);
- общее число операторов в программе;
- общее число операндов в программе.

Операторы метода OnLoginClicked класса LoginViewModel приведены в таблице 10.

Таблица 10 – Операторы

Операторы	Число вхождений
=	15
.	27
!=	3
==	3
New	4
Open	1
Close	1
ExecuteReader	1
Add	1
;	20
,	8
{}	9
n1 = 12	N1 = 93

Операнды метода OnLoginClicked класса LoginViewModel приведены в таблице 11.

Таблица 11 – Операнды

Операнды	Число вхождений
pwdlabel	1
loglabel	1
cn	5
cmd	4
l	2
reader	6
id	2
lo	2
pw	3
t	2
th	4
u	2
selected	2
n2 = 13	N2 = 36

Длина программы рассчитывается по формуле:

$$N = N1 + N2 = 93 + 36 = 129.$$

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 12 + 13 = 25.$$

Объем программы рассчитывается по формуле:

$$NV = N * \log_2(n) = 129 * 4.64 = 599.$$

Потенциальный объем программы

$$V^* = (2 + n) * \log_2(2 + n) = 4 * \log_2(4) = 8.$$

Теоретический уровень программы рассчитывается по формуле:

$$L = V^* / NV = 8 / 599 = 0.013.$$

Сложность программы рассчитывается по формуле:

$$HD = (n1 / 2) * (N2 / n2) = (12 / 2) * (93 / 13) = 43.$$

Значение уровня программы согласно метрике Холстеда равняется 0.013.

Уровень программы имеет значение ниже среднего по данным измерений из книги Холстеда [20].

4.4.4 Метрика Чепина

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода [17].

Все множество переменных, составляющих список ввода-вывода, разбивается на 4 функциональные группы:

- вводимые переменные для расчетов и для обеспечения вывода,
- модифицируемые, или создаваемые внутри программы переменные,
- переменные, участвующие в управлении работой программного модуля (управляющие переменные),
- не используемые в программе («паразитные») переменные.

Каждая переменная может выполнять несколько функций одновременно, поэтому необходимо учитывать ее в каждой соответствующей функциональной группе. Метрика Чепина рассчитывается следующим образом:

$$Q = a_1 * P + a_2 * M + a_3 * C + a_4 * T,$$

где a_1, a_2, a_3, a_4 – весовые коэффициенты.

Для отражения различного влияния на сложность программы каждой группы использованы весовые коэффициенты. Наибольший вес, равный 3, по мнению Чепина, имеет функциональная группа С, так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: $a_1=1$, $a_2=2$, $a_4=0.5$. Неиспользуемые переменные имеют ненулевой коэффициент. Несмотря на то, что они не увеличивают сложность программы, иногда они затрудняют ее понимание. В представленной в листинге 1 функции $P = 3$, $M = 10$, $C = 0$, $T = 0$ [17]. Учитывая весовые коэффициенты:

$$Q = 3 + 2*10 + 3*0 + 0.5*0 = 23.$$

Значение метрики Чепина равняется 23.

Если в результате изменения рассматриваемых функций значение метрики значительно увеличиться, это будет означать низкое качество изменений.

4.5 Планирование разработки и оценка бюджета

При помощи программы для управления проектами Microsoft Project 2007 было выполнено планирование разработки и оценка бюджета [21]. Проект был начат 12 апреля 2023 года.

При планировании разработки было выделено 22 задачи. Список задач и их описание представлено на рисунке 35.

Диаграмма Ганта, построенная на основе введенных данных в среде Microsoft Project 2007 приведена на рисунке Б.4 в приложении Б.

	Название задачи	Длительность	Начало	Окончание	Предшественники
1	Начало реализации проекта	0 дней	Ср 12.04.23	Ср 12.04.23	
2	Анализ требований	7 дней	Ср 12.04.23	Чт 20.04.23	
3	Анализ предметной области	2 дней	Ср 12.04.23	Чт 13.04.23	
4	Анализ аналогичных приложений	3 дней	Пт 14.04.23	Вт 18.04.23	3
5	Постановка задачи	1 день	Ср 19.04.23	Ср 19.04.23	4
6	Анализ функциональных требований	1 день	Чт 20.04.23	Чт 20.04.23	5
7	Анализ требований завершен	0 дней	Чт 20.04.23	Чт 20.04.23	6
8	Проектирование программных средств	8 дней	Пт 21.04.23	Вт 02.05.23	
9	Проектирование базы данных	2 дней	Пт 21.04.23	Пн 24.04.23	7
10	Проектирование архитектуры приложения	3 дней	Вт 25.04.23	Чт 27.04.23	9
11	Проектирование интерфейса приложения	3 дней	Пт 28.04.23	Вт 02.05.23	10
12	Проектирование программных средств завершено	0 дней	Вт 02.05.23	Вт 02.05.23	11
13	Разработка программных средств	17 дней	Ср 03.05.23	Чт 25.05.23	
14	Разработка базы данных	3 дней	Ср 03.05.23	Пт 05.05.23	12
15	Разработка клиентского приложения	14 дней	Пн 08.05.23	Чт 25.05.23	14
16	Разработка программных средств завершена	0 дней	Чт 25.05.23	Чт 25.05.23	15
17	Тестирование	5 дней	Пт 26.05.23	Чт 01.06.23	
18	Тестирование базы данных	1 день	Пт 26.05.23	Пт 26.05.23	16
19	Тестирование работы приложения	1 день	Пн 29.05.23	Пн 29.05.23	18
20	Исправление ошибок	3 дней	Вт 30.05.23	Чт 01.06.23	19
21	Тестирование завершено	0 дней	Чт 01.06.23	Чт 01.06.23	20
22	Конец проекта	0 дней	Чт 01.06.23	Чт 01.06.23	21

Рисунок 35 – Список задач проекта

Статистика затрат для проекта приведена на рисунке 36.

Статистика проекта для 'Проект1.mpr'			
	Начало		Окончание
Текущее	Ср 12.04.23		Чт 01.06.23
Базовое	НД		НД
Фактическое	НД		НД
Отклонение	0д		0д
	Длительность	Трудозатраты	Затраты
Текущие	37д	296ч	103 420,00 Р
Базовые	0д?	0ч	0,00 Р
Фактические	0д	0ч	0,00 Р
Оставшиеся	37д	296ч	103 420,00 Р
Процент завершения			
Длительность: 0%		Трудозатраты: 0%	
			Заккрыть

Рисунок 36 – Статистика проекта

Из полученного отчета видно, что предполагаемые затраты на разработку составляют 103 420 рублей, а общая длительность проекта – 37 дней.

Заключение

В ходе выполнения выпускной квалификационной работы разработано мобильное приложение – агрегатор аптечных товаров.

Выполнен анализ предметной области аптечных агрегаторов. Изучены информационные источники по данной теме, выделены наиболее популярные в настоящее время программные средства-агрегаторы.

Проведен сравнительный анализ программных средств-агрегаторов аптечных товаров, выявлены их достоинства и недостатки. На основе данного анализа сформулированы требования к разрабатываемому мобильному приложению. Создана диаграмма вариантов использования, описывающая основные функции мобильного приложения-агрегатора аптечных товаров.

Проведено планирование и расчет бюджета разрабатываемого программного средства при помощи системы управления проектами Microsoft Project 2007.

На этапе проектирования мобильного приложения определена его архитектура, разработана модель базы данных и спроектирован интерфейс приложения. Также были построены концептуальная и логическая модели базы данных и диаграмма классов UML.

На этапе реализации мобильного приложения была построена физическая модель базы данных, на основе которой база была создана и заполнена тестовыми данными. Была разработана диаграмма компонентов приложения.

Произведено функциональное тестирование разработанного программного обеспечения. Все тесты пройдены успешно, ошибок не выявлено.

Разработка программного обеспечения проводилась в среде разработки Microsoft Visual Studio 2022 на языке программирования C# в операционной системе Windows 10 с использованием библиотеки Xamarin.Forms.

Список использованных источников

1. Аптеки Вита, информация о компании [Электронный ресурс]. URL: <https://vitaexpress.ru/about/> (дата обращения: 15.04.2023).
2. О сервисе Apteka.ru [Электронный ресурс]. URL: <https://apteka.ru/about/> (дата обращения: 15.04.2023).
3. О компании СБЕР ЕАПТЕКА [Электронный ресурс]. URL: <https://www.eapteka.ru/penza/landing/about/> (дата обращения: 15.04.2023).
4. О проекте – Все аптеки [Электронный ресурс]. URL: <https://vseapteki.ru/about/> (дата обращения: 15.04.2023).
5. Информация о проекте Здравсити [Электронный ресурс]. URL: <https://zdravcity.ru/about/> (дата обращения: 15.04.2023).
6. О компании – Мегааптека.ру [Электронный ресурс]. URL: <https://megapteka.ru/about> (дата обращения: 15.04.2023).
7. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание / Д. Арлоу, А. Нейштадт – СПб: Символ – Плюс, 2007. – 624 с.
8. Г. Шилдт. Полный справочник по C# – Вильямс, 2004 – 748 с.
9. Что такое Xamarin? [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/xamarin/get-started/what-is-xamarin> (дата обращения: 20.04.2023).
10. Фаулер М. Архитектура корпоративных программных приложений. – Москва.: Вильямс, 2006. – ISBN 5-8459-0579-6.
11. Костюкова А. Базы данных: общий курс. Часть 1: учебное пособие. – Москва.: Академия Естествознания, 2018. – ISBN 978-5-91327-538-7.
12. Федорова К.П. Объектный подход в проектировании баз данных // Портал научно-практических публикаций [Электронный ресурс]. URL: <http://portalnp.ru/2014/06/1987> (дата обращения: 07.12.2022).
13. Якоб Нильсен, Хоа Лоранжер. Web-дизайн: удобство использования Web-сайтов = Prioritizing Web Usability. – М.: «Вильямс», 2007. – С. 368.

14. Куликов С. Тестирование программного обеспечения. Базовый курс (3-изд.). – Минск.: ЕРАМ Systems, 2021. – 300 с.

15. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. – Киев: ДиаСофт, 2001. – 544 с.

16. Тестирование мобильных приложений: типы, особенности – инструменты для тестирования приложений [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/testirovanie-mobilnyh-prilozhenii/> (дата обращения: 25.05.2023).

17. Метрики Кода И Их Практическая Реализация В Subversion И Clearcase. Часть 1 – Метрики [Электронный ресурс]. URL: <https://web.archive.org/web/20110913071425/http://www.rusarticles.com/programmy-statya/metriki-koda-i-ix-prakticheskaya-realizaciya-v-subversion-i-clearcase-chast-1-metriki-603328.html> (дата обращения: 25.05.23).

18. Аверьянов, А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.

19. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л.Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4. – С. 950 – 959.

20. Холстед М.Х. Начала науки о программах. – М.: Финансы и статистика, 1981. – 128с.

21. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. – 102 с.

Приложение А
АТТРИБУТЫ СУЩНОСТЕЙ БАЗЫ ДАННЫХ
(рекомендуемое)

Таблица А.1 – Пользователь

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_User	Номер покупателя в системе	Целый	6	1-999999	нет	РК	12	-
PNumber	Номер телефона покупателя	Символьный	11	'0-9'	нет		«+78005553535»	-
NOrders	Количество заказов	Целый	3	0-999	нет		3	-
Pass	Пароль	Символьный	30	«a-zA-Z0-9»	нет		password	-

Таблица А.2 – Заказ

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_Order	Номер лекарства в системе	Целый	6	1-999999	нет	РК	000111	-
Id_Concret Med	Идентификатор товара	Целый	6	1-999999	нет		23	-
Number	Количество	Целый	3	1-999	нет		3	
Id_Place	Идентификатор пункта выдачи	Целый	6	1-999999	нет		3	-

Таблица А.3 – Лекарство

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_Med	Идентификатор лекарства в системе	Целый	6	1-999999	нет	РК	9	-
Name	Название лекарства	Дата/время	8	'01.01.2022 – 31.12.2099'	нет		“Кагоцел”	-
Image	url-адрес изображения	Символьный	200	'a-zA-Z0-9_-./'	нет		“https://f.stolichki.ru/10683.jpg”	-
Id_Form	Идентификатор формы выпуска	Целый	6	1-999999	нет		1	-
ID_Active	Идентификатор действующего вещества	Целый	6	1-999999	нет		5	
ID_Cat	Идентификатор категории препарата	Целый	6	1-999999	нет		2	

Таблица А.4 – Товар

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_ConMed	Идентификатор конкретного лекарства в системе	Целый	6	1-999999	нет	РК	5	-
Id_Med	Идентификатор лекарства в системе	Целый	6	1-999999	нет		2	-
Id_Pharm	Идентификатор аптечной сети	Целый	6	1-999999	нет		4	-
Cost	Стоимость товара	число с плавающей точкой	6	0000,00-9999,99	нет		37.9	-

Таблица А.5 – Действующее вещество

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_Active	Номер вещества в системе	Целый	1	0-9	нет	РК	1	-
Active	Наименование вещества	Символьный	50	‘а-я’, ‘А-Я’, ‘ ’	нет		“Ксилотазолин”	-

Таблица А.6 – Категория

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_Category	Номер категории препарата в системе	Целый	1	0-9	нет	РК	1	-
Category	Наименование категории	Символьный	50	‘а-я’, ‘А-Я’, ‘ ’	нет		“Жаропонижающее”	-

Таблица А.7 – Форма выпуска

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_Form	Номер формы выпуска в системе	Целый	1	0-9	нет	РК	1	-
Form	Форма выпуска	Символьный	50	‘а-я’, ‘А-Я’, ‘ ’	нет		“Таблетки”	-

Таблица А.8 – Аптечная сеть

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_Pharma	Идентификатор аптечной сети	Целый	1	0-9	нет	РК	1	-
Pharma	Название	Символьный	100	‘а-яА-Я’, ‘,’	нет		“Аптека Апрель”	-

Таблица А.9 – Адрес

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID_Place	Номер пункта выдачи в системе	Целый	1	0-9	нет	РК	1	-
Place	Адрес пункта выдачи	Символьный	100	‘а-яА-Я’, ‘,’	нет		«Г. Пенза, ул. Мира, д. 6»	-

Приложение Б
ГРАФИЧЕСКАЯ ЧАСТЬ
(рекомендуемое)

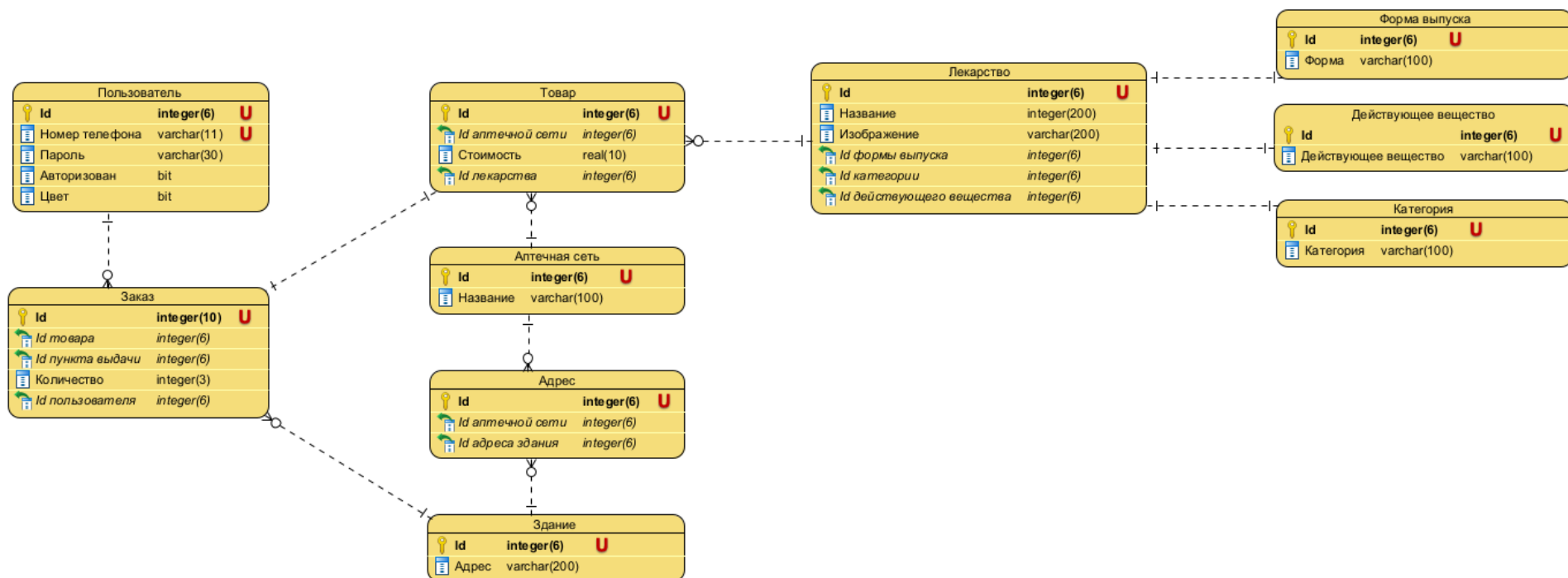


Рисунок Б.1 – Логическая модель базы данных предметной области «Аптека»

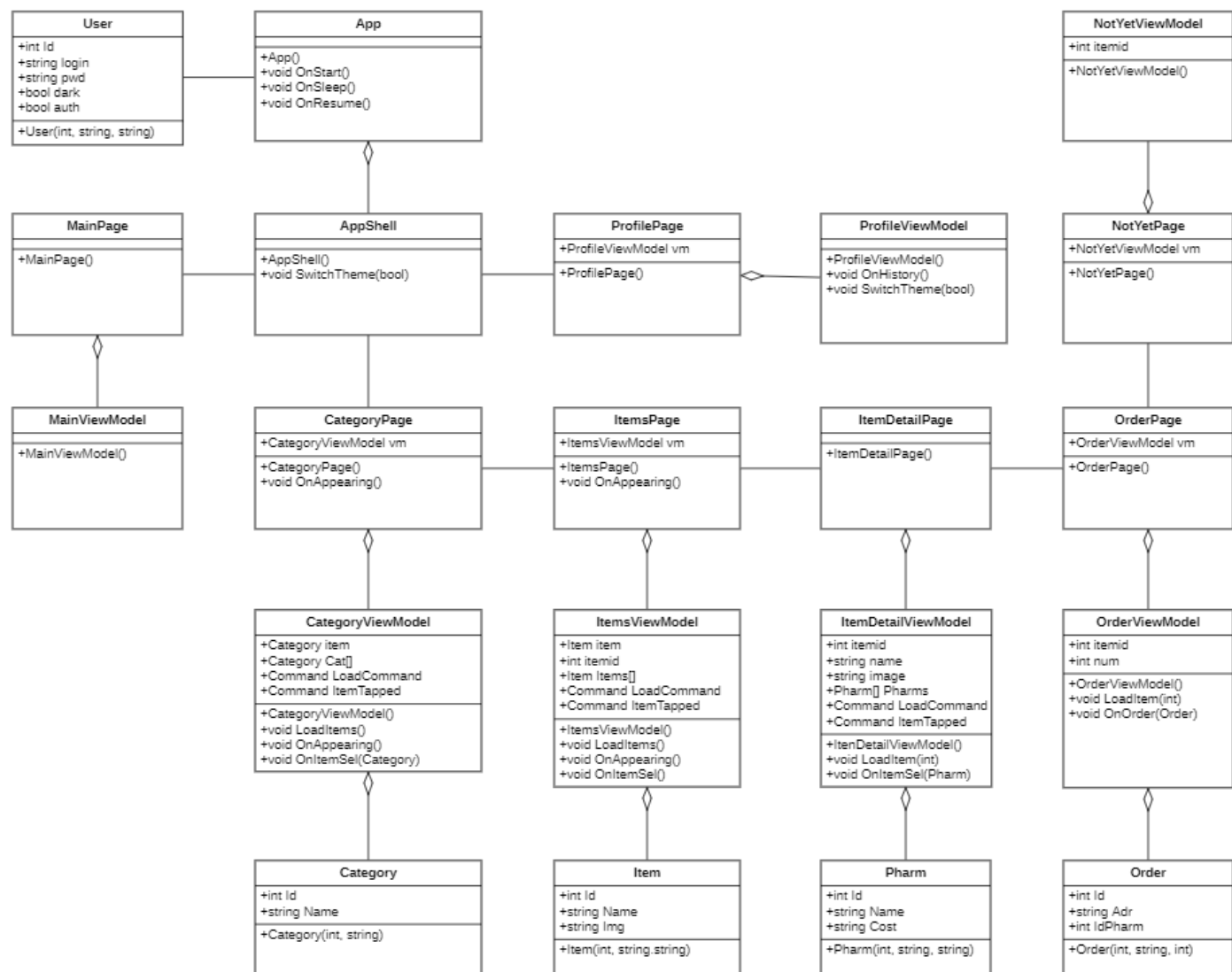


Рисунок Б.2 – Диаграмма классов

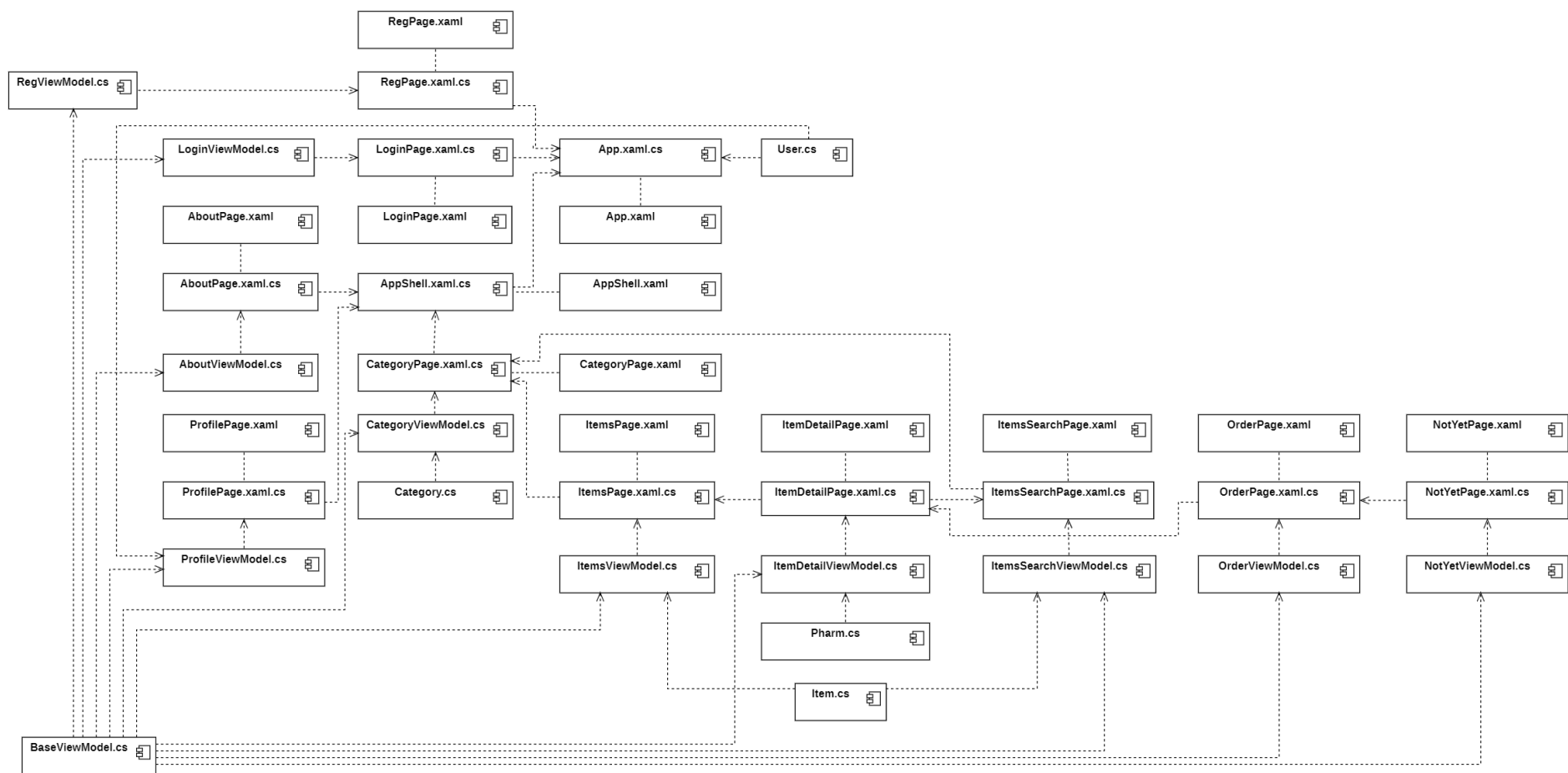


Рисунок Б.3 – Диаграмма компонентов

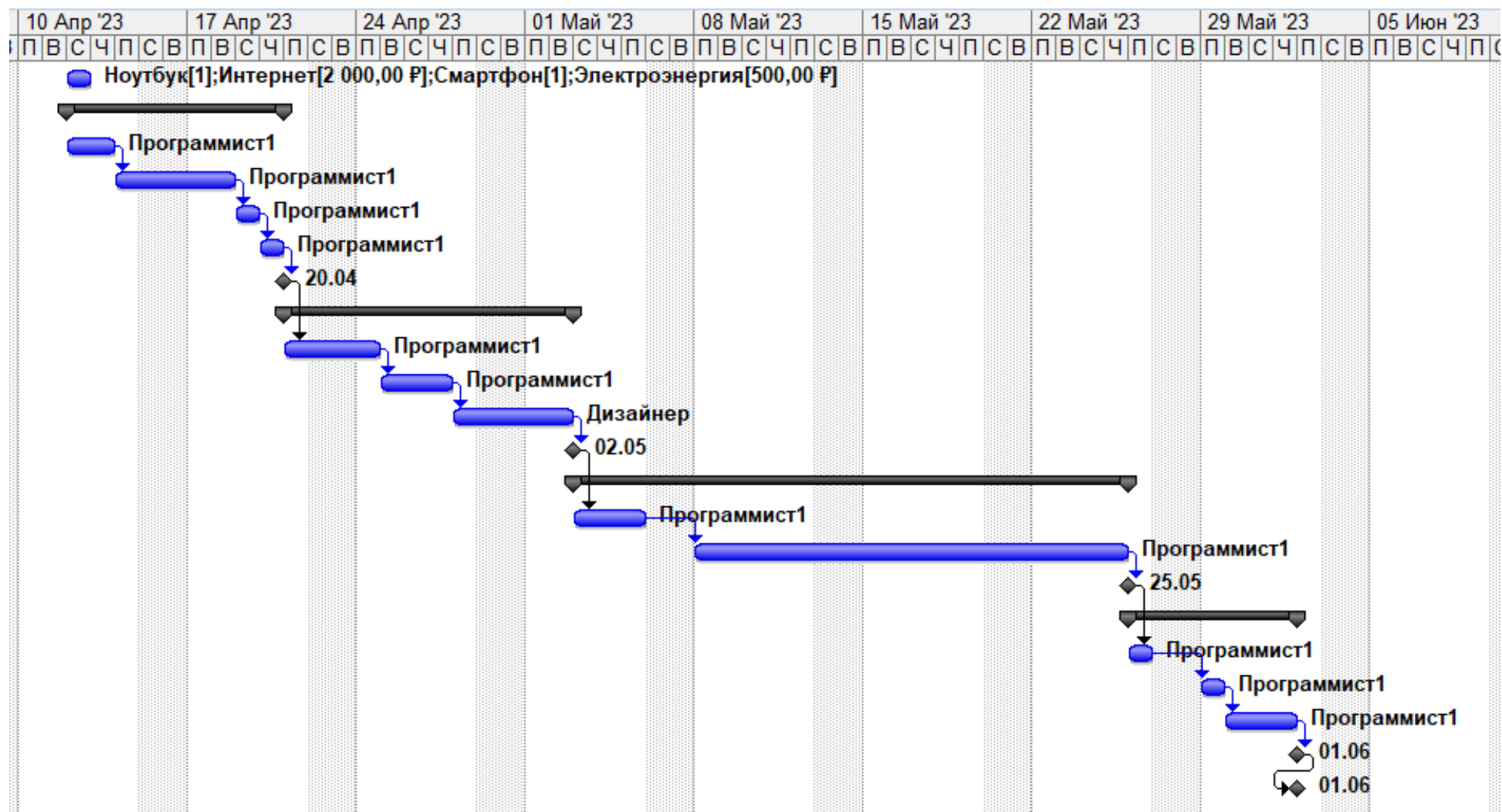


Рисунок Б.4 – Диаграмма Ганта

Приложение В
СКРИПТ ДЛЯ СОЗДАНИЯ БАЗЫ ДАННЫХ
(рекомендуемое)

```
CREATE TABLE Adress (id_adr int IDENTITY NOT NULL,  
    adress varchar(150) NOT NULL,  
    PRIMARY KEY (id_adr));
```

```
CREATE TABLE Pharma (id_pharm int IDENTITY NOT NULL,  
    namePh varchar(100) NOT NULL,  
    PRIMARY KEY (id_pharm));
```

```
CREATE TABLE Place (id_place int IDENTITY NOT NULL,  
    id_pharm int NOT NULL,  
    id_adr int NOT NULL,  
    PRIMARY KEY (id_place));
```

```
CREATE TABLE Med (id_med int IDENTITY NOT NULL,  
    id_activ int NOT NULL,  
    id_form int NOT NULL,  
    img varchar(200) NOT NULL,  
    nameMed varchar(200) NOT NULL,  
    id_cat int NOT NULL,  
    PRIMARY KEY (id_med));
```

```
CREATE TABLE ConcretMed (id_conMed int IDENTITY NOT NULL,  
    id_pharm int NOT NULL,  
    cost real NOT NULL,  
    id_med int NOT NULL,  
    PRIMARY KEY (id_conMed));
```

```
CREATE TABLE ActiveSub (id_activ int IDENTITY NOT NULL,  
    activeSub varchar(100) NOT NULL,  
    PRIMARY KEY (id_activ));
```

```
CREATE TABLE Form (id_form int IDENTITY NOT NULL,  
    form varchar(100),  
    PRIMARY KEY (id_form));
```

```
CREATE TABLE Category (id_cat int IDENTITY NOT NULL,  
    category varchar(100) NOT NULL,  
    PRIMARY KEY (id_cat));
```

```
ALTER TABLE Place ADD CONSTRAINT FKPlace316558 FOREIGN KEY  
    (id_adr) REFERENCES Adress (id_adr);
```

```
ALTER TABLE Place ADD CONSTRAINT FKPlace468674 FOREIGN KEY  
    (id_pharm) REFERENCES Pharma (id_pharm);
```

```
ALTER TABLE Med ADD CONSTRAINT FKMed949908 FOREIGN KEY (id_cat)  
    REFERENCES Category (id_cat);
```

```
ALTER TABLE Med ADD CONSTRAINT FKMed450364 FOREIGN KEY (id_activ)  
    REFERENCES ActiveSub (id_activ);
```

```
ALTER TABLE Med ADD CONSTRAINT FKMed886203 FOREIGN KEY (id_form)  
    REFERENCES Form (id_form);
```

```
ALTER TABLE ConcretMed ADD CONSTRAINT FKConcretMed951047 FOREIGN  
    KEY (id_med) REFERENCES Med (id_med);
```

```
ALTER TABLE ConcretMed ADD CONSTRAINT FKConcretMed491945 FOREIGN  
    KEY (id_pharm) REFERENCES Pharma (id_pharm);
```

Приложение Г
СКРИПТЫ ХРАНИМЫХ ПРОЦЕДУР
(рекомендуемое)

Листинг Г.1 – Процедура поиска адреса аптеки

```
USE [PHARMA]
CREATE PROCEDURE FindApt
    @I int
AS
BEGIN
    SELECT Pharma.id_pharm,
        Pharma.namePh,
        Adress.adress
    from Adress, Place, Pharma
    where Adress.id_adr=Place.id_adr and
        Place.id_pharm=Pharma.id_pharm and
        Pharma.id_pharm=@I
END
```

Листинг Г.2 – Процедура поиска зарегистрированного пользователя

```
USE [PHARMA]
CREATE PROCEDURE FindUser
    @l varchar
AS
BEGIN
    SELECT Users.id_user,
        Users.phone,
        Users.pwd,
        Users.theme
    from Users
    where Users.phone=@l
END
```

Листинг Г.3 – Процедура получения списка категорий товаров

```
USE [PHARMA]
CREATE PROCEDURE GetCats
AS
BEGIN
    SELECT Category.id_cat,
        Category.category
    from Category
END
```

Листинг Г.4 – Процедура получения информации о товаре

```
USE [PHARMA]
CREATE PROCEDURE GetMedInfo
    @I int
AS
BEGIN
    SELECT Med.nameMed,
```

```

Med.img,
Pharma.namePh,
ConcretMed.cost,
Pharma.id_pharm
from ConcretMed, Med, Pharma
where Med.id_med=ConcretMed.id_med and
      ConcretMed.id_pharm=Pharma.id_pharm and
      Med.id_med=@I
END

```

Листинг Г.5 – Процедура получения списка товаров из конкретной категории

```

USE [PHARMA]
CREATE PROCEDURE GetMedsFromCat
    @C int
AS
BEGIN
    SELECT Med.id_med,
           Med.nameMed,
           Med.img
    from Category, Med
    where Med.id_cat=Category.id_cat and
           Category.id_cat=@C
END

```

Приложение Д
ЛИСТИНГ МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Д.1 – Файл AppShell.xaml.cs

```

using App6.Models;
using App6.Themes;
using App6.ViewModels;
using App6.Views;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using Xamarin.Forms;
namespace App6
{
    public partial class AppShell : Xamarin.Forms.Shell
    {
        public AppShell()
        {
            InitializeComponent();
            Routing.RegisterRoute(nameof(ItemsPage),
                typeof(ItemsPage));
            Routing.RegisterRoute(nameof(ItemsSearchPage),
                typeof(ItemsSearchPage));
            Routing.RegisterRoute(nameof(ItemDetailPage),
                typeof(ItemDetailPage));
            Routing.RegisterRoute(nameof(OrderPage),
                typeof(OrderPage));
            Routing.RegisterRoute(nameof(NotYetPage),
                typeof(NotYetPage));
            Routing.RegisterRoute(nameof(LoginPage),
                typeof(LoginPage));
            Routing.RegisterRoute(nameof(RegPage),
                typeof(RegPage));
        }
        public static void SwitchTheme(bool e)
        {
            ICollection<ResourceDictionary>mergedDictionaries =
                Application.Current.Resource.MerDictionaries;
            if (mergedDictionaries != null){
                switch (e){
                    case true:
                        mergedDictionaries.Add(new DarkTheme());
                        break;
                    case false:
                        mergedDictionaries.Add(new LightTheme());
                        break;
                    default:
                        break;
                }
            }
        }
    }
}

```

Листинг Д.2 – Файл AppShell.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<Shell xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  xmlns:local="clr-namespace:App6.Views"
  Title="App6"
  BackgroundColor="{DynamicResource PageBackgroundColor}"
  x:Class="App6.AppShell">
  <Shell.Resources>
    <ResourceDictionary Source="Themes/LightTheme.xaml">
      <Style x:Key="TabStyle"
        TargetType="Element">
        <Setter Property="Shell.BackgroundColor"
          Value="{DynamicResource NavigationBarColor}"/>
        <Setter Property="Shell.ForegroundColor"
          Value="{DynamicResource PageBackgroundColor}"/>
        <Setter Property="Shell.TitleColor"
          Value="White"/>
        <Setter Property="Shell.DisabledColor"
          Value="{DynamicResource NavigationBarColor_dis}"/>
        <Setter Property="Shell.UnselectedColor"
          Value="{DynamicResource NavigationBarColor_dis}"/>
        <Setter Property="Shell.TabBarBackgroundColor"
          Value="{DynamicResource NavigationBarColor}"/>
        <Setter Property="Shell.TabBarForegroundColor"
          Value="{DynamicResource NavigationBarColor}"/>
        <Setter Property="Shell.TabBarUnselectedColor"
          Value="{DynamicResource NavigationBarColor_abl}"/>
        <Setter Property="Shell.TabBarTitleColor"
          Value="{DynamicResource NavigationBarColor_dis}"/>
      </Style>
      <Style TargetType="TabBar"
        BasedOn="{StaticResource TabStyle}"/>
      <Style x:Key="Navstyle"
        TargetType="NavigationPage">
        <Setter Property="Shell.BackgroundColor"
          Value="{DynamicResource PageBackgroundColor}"/>
        <Setter Property="Shell.ForegroundColor"
          Value="{DynamicResource TextColor}"/>
        <Setter Property="Shell.TitleColor"
          Value="White"/>
      </Style>
    </ResourceDictionary>
  </Shell.Resources>
  <ShellContent Title="log"
    Icon="icon_about.png"
    Route="LoginPage"
    ContentTemplate="{DataTemplate local:LoginPage}"/>
  <TabBar>
    <ShellContent Title="Главная"
```

```

        Icon="icon_main.png"
        Route="AboutPage"
        ContentTemplate="{DataTemplate local:AboutPage}"/>
<ShellContent Title="Поиск"
        Icon="icon_search.png"
        Route="CategoryPage"
        ContentTemplate="{DataTemplate
local:CategoryPage}"/>
<ShellContent Title="Профиль"
        Icon="icon_user.png"
        Route="ProfilePage"
        ContentTemplate="{DataTemplate local:ProfilePage}"/>
</TabBar>
</Shell>

```

Листинг Д.3 – Файл LoginPage.xaml.cs

```

using App6.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace App6.Views{
    public partial class LoginPage : ContentPage{
        LoginViewModel _viewModel;
        public LoginPage(){
            InitializeComponent();
            this.BindingContext = _viewModel = new
                LoginViewModel();
        }
        private void loginEntry_Completed(object sender,
            EventArgs e){
            _viewModel.log = loginEntry.Text;
            string ss = loginEntry.Text;
            string strPattern = "([0-9])";
            MatchCollection mm = Regex.Matches(ss, strPattern);
            _viewModel.loglabel = "";
            if (ss.Length == 0){
                _viewModel.loglabel = "Логин не может быть
                    пустым!";
            }
            if (mm.Count == 0){
                _viewModel.loglabel += "Логин может содержать
                    только цифры!";
            }
            if (ss.Length < 11){
                _viewModel.loglabel += "Логин должен состоять
                    из 11 символов!";
            }
        }
    }
}

```

```

    }
    loginLabel.Text = _viewModel.loglabel;
}
private void passwordEntry_Completed(object sender,
EventArgs e){
    _viewModel.pwd = passwordEntry.Text;
    string ss = passwordEntry.Text;
    string strPattern = @"^[a-zA-Za-яA-Я0-9]*$";
    MatchCollection mm = Regex.Matches(ss, strPattern);
    _viewModel.pwdlabel = "";
    if (ss.Length == 0){
        _viewModel.pwdlabel = "Пароль не может быть
        пустым!";
    }
    if (mm.Count == 0){
        _viewModel.pwdlabel += "Пароль не может
        содержать недопустимые символы!";
    }
    if (ss.Length < 6){
        _viewModel.pwdlabel += "Пароль должен состоять
        минимум из 6 символов!";
    }
    passwordLabel.Text= _viewModel.pwdlabel;
}
}
}

```

Листинг Д.4 – Файл LoginPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:d="http://xamarin.com/schemas/2014/forms/design"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
        compatibility/2006"
    xmlns:vm="clr-namespace:App6.ViewModels"
    x:DataType="vm:LoginViewModel"
    mc:Ignorable="d"
    x:Class="App6.Views.LoginPage"
    Shell.NavBarIsVisible="False">
    <ContentPage.Content>
        <StackLayout Padding="10,0,10,0"
            VerticalOptions="Center"
            Spacing="50">
            <StackLayout >
                <Entry Placeholder="Номер телефона"
                    PlaceholderColor="#905090"
                    MaxLength="11"
                    x:Name="loginEntry"
                    TextChanged="loginEntry_Completed"/>
                <Label TextColor="red"
                    Text="{Binding Loglabel}"

```

```

        x:Name="loginLabel"/>
</StackLayout>
<StackLayout>
    <Entry Placeholder="Пароль"
        PlaceholderColor="#905090"
        MaxLength="30"
        IsPassword="true"
        x:Name="passwordEntry"
        TextChanged="passwordEntry_Completed"/>
    <Label TextColor="red"
        Text="{Binding Pwdlabel}"
        x:Name="passwordLabel"/>
</StackLayout>
<Button VerticalOptions="Center"
    Text="Войти"
    Command="{Binding LoginCommand}"/>
<StackLayout Padding="10,0,10,0"
    VerticalOptions="Center"
    Spacing="10">
    <Label Text="Еще не зарегистрированы?"
        FontSize="16"
        VerticalOptions="CenterAndExpand"
        HorizontalOptions="Center"/>
    <Button VerticalOptions="Center"
        Text="Зарегистрироваться"
        Command="{Binding RegCommand}"/>
</StackLayout>
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

Листинг Д.5 – Файл LoginViewModel.cs

```

using App6.Models;
using App6.Views;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Text;
using Xamarin.Forms;
namespace App6.ViewModels{
    public class LoginViewModel : BaseViewModel{
        public string log;
        public string pwd;
        public static User selected;
        public Command LoginCommand { get; }
        public Command RegCommand { get; }
        public string loglabel;
        public string pwdlabel;
        public string Loglabel{
            get{
                return loglabel;
            }
        }
    }
}

```

```

    }
    set{
        loglabel = value;
    }
}
public string Pwdlabel{
    get{
        return pwdlabel;
    }
    set{
        pwdlabel = value;
    }
}
public LoginViewModel(){
    LoginCommand = new Command(OnLoginClicked);
    RegCommand = new Command(OnRegClicked);
    loglabel = "";
    pwdlabel = "";
}
private async void OnLoginClicked(object obj){
    if ((pwdlabel != "") || (loglabel != "")) return;
    try{
        SqlConnection cn = new
            SqlConnection("workstation
            id=KSENYAPHARMA.mssql.somee.com; packet
            size=4096; user id=kkk_SQLLogin_2;
            pwd=ydk4sfi7ym; data
            source=KSENYAPHARMA.mssql.somee.com;
            persist security info=False; initial
            catalog=KSENYAPHARMA");
        if (cn.State ==
            System.Data.ConnectionState.Closed){
            cn.Open();
            SqlCommand cmd = new
                SqlCommand("FindUser", cn);
            cmd.CommandType =
                System.Data.CommandType.StoredProcedure;
            SqlParameter l = new SqlParameter{
                ParameterName = "@l",
                Value = log
            };
            cmd.Parameters.Add(l);
            SqlDataReader reader =
                cmd.ExecuteReader();
            while (reader.Read()){
                object id = reader.GetValue(0);
                object lo = reader.GetValue(1);
                object pw = reader.GetValue(2);
                object t = reader.GetValue(3);
                string th;

```

```

        if (Int32.Parse(t.ToString()) == 1)
            th = "";
        else th = "";
        if (pw.ToString() == pwd){
            User u = new
                User(Int32.Parse(id.ToStrin
                    g()), lo.ToString(),
                    pw.ToString(), th, 1);
            selected = u;
        }
    }
    cn.Close();
}
}
catch (Exception ex){
    Console.WriteLine(ex.Message);
}
if (selected != null){
    await
        Shell.Current.GoToAsync($"{nameof(AboutP
            age)}");
}
}
private async void OnRegClicked(object obj){
    await
        Shell.Current.GoToAsync($"{nameof(RegPage)}?{na
            meof(RegViewModel.ItemId)}={1}");
}
}
}

```

Листинг Д.6 – Файл RegPage.xaml.cs

```

using App6.ViewModels;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace App6.Views{
    public partial class RegPage : ContentPage{
        RegViewModel _viewModel;
        public RegPage(){
            InitializeComponent();
            this.BindingContext = _viewModel = new
                RegViewModel();
        }
    }
}

```

```

private void loginEntry_Completed(object sender,
    EventArgs e){
    _viewModel.log = loginEntry.Text;
    string ss = loginEntry.Text;
    string strPattern = "([0-9])";
    MatchCollection mm = Regex.Matches(ss, strPattern);
    _viewModel.loglabel = "";
    if (ss.Length == 0){
        _viewModel.loglabel = "\nЛогин не может быть
            пустым!";
    }
    if (mm.Count != ss.Length){
        _viewModel.loglabel += "\nЛогин может содержать
            только цифры!";
    }
    if (ss.Length < 11){
        _viewModel.loglabel += "\nЛогин должен состоять
            из 11 символов!";
    }
    else{
        SqlConnection cn = new
            SqlConnection("workstation
                id=KSENYAPHARMA.mssql.somee.com; packet
                size=4096; user id=kkk_SQLLogin_2;
                pwd=ydk4sfi7ym; data
                source=KSENYAPHARMA.mssql.somee.com;
                persist security info=False; initial
                catalog=KSENYAPHARMA");
        bool flag = false;
        if (cn.State ==
            System.Data.ConnectionState.Closed){
            cn.Open();
            SqlCommand cmd = new
                SqlCommand("FindUser", cn);
            cmd.CommandType =
                System.Data.CommandType.StoredProcedure;
            SqlParameter l = new SqlParameter{
                ParameterName = "@l",
                Value = ss
            };
            cmd.Parameters.Add(l);
            SqlDataReader reader =
                cmd.ExecuteReader();
            while (reader.Read()){
                flag = true;
            }
        }
        if (flag) _viewModel.loglabel += "\nЛогин уже
            занят!";
    }
    loginLabel.Text = _viewModel.loglabel;
}

```

```

    }
    private void passwordEntry_Completed(object sender,
        EventArgs e){
        _viewModel.pwd = passwordEntry.Text;
        string ss = passwordEntry.Text;
        string strPattern = @"^[a-zA-Za-яA-Я0-9]*$";
        MatchCollection mm = Regex.Matches(ss, strPattern);
        _viewModel.pwdlabel = "";
        if (ss.Length == 0){
            _viewModel.pwdlabel = "\nПароль не может быть
                пустым!";
        }
        if (mm.Count == 0){
            _viewModel.pwdlabel += "\nПароль не может
                содержать недопустимые символы!";
        }
        if (ss.Length < 6){
            _viewModel.pwdlabel += "\nПароль должен
                состоять минимум из 6 символов!";
        }
        passwordLabel.Text = _viewModel.pwdlabel;
    }
}

```

Листинг Д.7 – Файл RegViewModel.cs

```

using App6.Models;
using App6.Views;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Text;
using Xamarin.Forms;
namespace App6.ViewModels
{
    public class RegViewModel : BaseViewModel
    {
        public Command RegCommand { get; }
        public string log;
        public string pwd;
        private int itemId;
        public string loglabel;
        public string pwdlabel;
        public RegViewModel(){
            RegCommand = new Command(OnRegClicked);
        }
        public int ItemId{
            get{
                return itemId;
            }
            set{

```

```

        itemId = value;
    }
}
public string Loglabel{
    get{
        return loglabel;
    }
    set{
        loglabel = value;
    }
}
public string Pwdlabel{
    get{
        return pwdlabel;
    }
    set{
        pwdlabel = value;
    }
}
private async void OnRegClicked(object obj){
    if ((pwdlabel != "") || (loglabel != "")) return;
    try{
        SqlConnection cn = new
            SqlConnection("workstation
            id=KSENYAPHARMA.mssql.somee.com; packet
            size=4096; user id=kkk_SQLLogin_2;
            pwd=ydk4sfi7ym; data
            source=KSENYAPHARMA.mssql.somee.com;
            persist security info=False; initial
            catalog=KSENYAPHARMA");
        if (cn.State ==
            System.Data.ConnectionState.Closed){
            cn.Open();
            SqlCommand cmd = new SqlCommand("Insert
            into Users (phone, pwd, theme, auth)
            VALUES (@phone, @pwd, @theme,
            @auth)", cn);
            cmd.CommandType =
                System.Data.CommandType.Text;
            cmd.Parameters.AddWithValue("@phone",
                log);
            cmd.Parameters.AddWithValue("@pwd", pwd);
            cmd.Parameters.AddWithValue("@theme", 0);
            cmd.Parameters.AddWithValue("@auth", 1);
            cmd.ExecuteScalar();
            cn.Close();
        }
    }
    catch (Exception ex){
        Console.WriteLine(ex.Message);
    }
}

```

```

        await
            Shell.Current.GoToAsync($"{nameof(AboutPage)}");
    }
}

```

Листинг Д.8 – Файл MainPage.xaml.cs

```

using App6.Models;
using App6.ViewModels;
using System;
using System.ComponentModel;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace App6.Views
{
    public partial class AboutPage : ContentPage
    {
        public AboutPage()
        {
            InitializeComponent();
        }
        private void Button_Clicked(object sender, EventArgs e)
        {
            Shell.Current.GoToAsync($"{nameof(CategorPage)}");
        }
    }
}

```

Листинг Д.9 – Файл ProfilePage.xaml.cs

```

using App6.Themes;
using App6.ViewModels;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using Xamarin.Forms;
using Xamarin.Forms.PlatformConfiguration;
using Xamarin.Forms.Xaml;
namespace App6.Views
{
    public partial class ProfilePage : ContentPage
    {
        ProfileViewModel _viewModel;
        public ProfilePage() {
            InitializeComponent();
            BindingContext = _viewModel = new
                ProfileViewModel();
        }
    }
}

```

```

        private void Switch_Toggled(object sender,
            ToggledEventArgs e){
            AppShell.SwitchTheme(e.Value);
        }
    }
}

```

Листинг Д.10 – Файл ProfilePage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App6.Views.ProfilePage"
    xmlns:vm="clr-namespace:App6.ViewModels"
    Title="{Binding Title}"
    BackgroundColor="{DynamicResource PageBackgroundColor}">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <ScrollView Grid.Row="0">
            <StackLayout Orientation="Vertical"
                Padding="30,24,30,24"
                Spacing="50" >
                <StackLayout Orientation="Horizontal"
                    Spacing="75">
                    <Label TextColor="{DynamicResource TextColor}"
                        Text="Номер телефона:"
                        FontSize="16"/>
                    <Label Text="{Binding Log}"
                        TextColor="{DynamicResource TextColor}"
                        FontSize="16"/>
                </StackLayout>
                <StackLayout Orientation="Horizontal"
                    Spacing="120">
                    <Label TextColor="{DynamicResource TextColor}"
                        Text="Город:"
                        VerticalOptions="CenterAndExpand"
                        HorizontalOptions="StartAndExpand"
                        FontSize="16"/>
                    <Picker WidthRequest="130"
                        FontSize="16"
                        Title="Выберите город:"
                        TitleColor="{DynamicResource TextColor}"
                        BackgroundColor="{DynamicResource
                            PageBackgroundColor}"
                        TextColor="{DynamicResource TextColor}">
                        <Picker.Items>
                            <x:String>Пенза</x:String>
                        </Picker.Items>
                    </Picker>
                </StackLayout>
            </ScrollView>
        </Grid>
    </ContentPage>

```

```

</StackLayout>
<StackLayout Orientation="Horizontal"
    Spacing="60">
    <Label TextColor="{DynamicResource TextColor}"
        Text="Накоплено бонусов:"
        FontSize="16"/>
    <Label Text="0"
        TextColor="{DynamicResource TextColor}"
        FontSize="16"/>
</StackLayout>
<StackLayout Orientation="Horizontal"
    Spacing="160">
    <Label Text="Темная тема"
        TextColor="{DynamicResource TextColor}"
        FontSize="16"/>
    <Switch Toggled="Switch_Toggled"
        OnColor="{DynamicResource NavigationBarColor}"
        ThumbColor="{DynamicResource
            NavigationBarColor}"/>
</StackLayout>
<Label Text="История заказов"
    TextColor="{DynamicResource TextColor}"
    FontSize="16">
    <Label.GestureRecognizers>
    <TapGestureRecognizer NumberOfTapsRequired="1"
        Command="{Binding Source={RelativeSource
            AncestorType={x:Type
                vm:ProfileViewModel}}}, Path=ItemTapped}"
        CommandParameter="{Binding .}"/>
    </TapGestureRecognizer>
    </Label.GestureRecognizers>
</Label>
<Label Text="Выход"
    TextColor="{DynamicResource TextColor}"
    FontSize="16" >
    <Label.GestureRecognizers>
    <TapGestureRecognizer NumberOfTapsRequired="1"
        Command="{Binding Source={RelativeSource
            AncestorType={x:Type
                vm:ProfileViewModel}}}, Path=ItemTapped1}"
        CommandParameter="{Binding .}"/>
    </TapGestureRecognizer>
    </Label.GestureRecognizers>
</Label>
</StackLayout>
</ScrollView>
</Grid>
</ContentPage>

```

Листинг Д.11 – Файл ProfileViewModel.cs

```

using App6.Models;
using App6.Views;
using System;
using System.Collections.Generic;
using System.Windows.Input;
using Xamarin.Essentials;
using Xamarin.Forms;
using App6.Themes;
using System.Data.SqlClient;
namespace App6.ViewModels
{
    public class ProfileViewModel : BaseViewModel
    {
        public User sel;
        public string log;
        public Command ItemTapped { get; }
        public Command ItemTapped1 { get; }
        public ProfileViewModel() {
            Title = "Профиль";
            ItemTapped = new Command(OnItemSelected);
            ItemTapped1 = new Command(OnItem1Selected);
            sel = LoginViewModel.selected;
            log = LoginViewModel.selected.Name;
        }
        public string Log{
            get
            {
                return log;
            }
            set
            {
                log = value;
            }
        }
        async void OnItemSelected() {
            await
                Shell.Current.GoToAsync($"{nameof(NotYetPage)}?
                {nameof(NotYetViewModel.ItemId)}={0}");
        }
        async void OnItem1Selected() {
            await Shell.Current.GoToAsync("//LoginPage");
        }
    }
}

```

Листинг Д.12 – Файл CategoryPage.xaml.cs

```

using App6.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```



```

        <Label Grid.Column="0"
            Text="{Binding Text}"
            Padding="10"
            TextColor="{DynamicResource TextColor}"
            LineBreakMode="WordWrap"
            Margin="0,10,0,0"
            VerticalOptions="CenterAndExpand"
            HorizontalOptions="StartAndExpand"
            Style="{DynamicResource
                ListItemTextStyle}"
            FontSize="16"/>
        <Grid.GestureRecognizers>
            <TapGestureRecognizer
                NumberOfTapsRequired="1"
                Command="{Binding
                    Source={RelativeSource
                        AncestorType={x:Type
                            local:CategoryViewModel}},
                    Path=ItemTapped}"
                CommandParameter="{Binding .}">
            </TapGestureRecognizer>
        </Grid.GestureRecognizers>
    </Grid>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</RefreshView>
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

Листинг Д.14 – Файл CategoryViewModel.cs

```

using App6.Models;
using App6.Views;
using System;
using System.Collections.ObjectModel;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Threading.Tasks;
using Xamarin.Forms;
namespace App6.ViewModels
{
    public class CategoryViewModel : BaseViewModel{
        private Category _selectedItem;
        public ObservableCollection<Category> Cats { get; }
        public Command LoadItemsCommand { get; }
        public Command AddItemCommand { get; }
        public Command<Category> ItemTapped { get; }
        public CategoryViewModel(){
            Title = "Категории";
            Cats = new ObservableCollection<Category>();

```

```

LoadItemsCommand = new Command(() =>
    ExecuteLoadItemsCommand());
ItemTapped = new Command<Category>(OnItemSelected);
}
void ExecuteLoadItemsCommand() {
    IsBusy = true;
    try{
        Cats.Clear();
        SqlConnection cn = new
            SqlConnection("workstation
                id=KSENYAPHARMA.mssql.somee.com; packet
                size=4096; user id=kkk_SQLLogin_2;
                pwd=ydk4sfi7ym; data
                source=KSENYAPHARMA.mssql.somee.com;
                persist security info=False; initial
                catalog=KSENYAPHARMA");
        if (cn.State ==
            System.Data.ConnectionState.Closed) {
            cn.Open();
            SqlCommand cmd = new SqlCommand("Select *
                from Category", cn);
            cmd.CommandType =
                System.Data.CommandType.Text;
            SqlDataReader reader =
                cmd.ExecuteReader();
            while (reader.Read()) {
                object id = reader.GetValue(0);
                object name = reader.GetValue(1);
                Category C = new
                    Category(Int32.Parse(id.ToString
                        ()), name.ToString());
                Cats.Add(C);
            }
            cn.Close();
        }
    }
    catch (Exception ex) {
        Console.WriteLine(ex.Message);
    }
    finally {
        IsBusy = false;
    }
}
public async void ExecuteLoadItemsSearchCommand(string
    search) {
    await
        Shell.Current.GoToAsync($"{nameof(ItemsSearchPa
            ge)}?{nameof(ItemsSearchViewModel.Search)}={sea
                rch}");
}
public void OnAppearing() {
    IsBusy = true;
}

```

```

        SelectedItem = null;
    }
    public Category SelectedItem{
        get => _selectedItem;
        set{
            SetProperty(ref _selectedItem, value);
            OnItemSelected(value);
        }
    }
    async void OnItemSelected(Category item){
        if (item == null)
            return;
        await
            Shell.Current.GoToAsync($"{nameof(ItemsPage)}?{
                nameof(ItemsViewModel.ItemId)}={item.Id}");
    }
}

```

Листинг Д.15 – Файл ItemsPage.xaml.cs

```

using App6.Models;
using App6.ViewModels;
using App6.Views;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace App6.Views{
    public partial class ItemsPage : ContentPage{
        ItemsViewModel _viewModel;
        public ItemsPage(){
            InitializeComponent();
            BindingContext = _viewModel = new ItemsViewModel();
        }
        protected override void OnAppearing(){
            base.OnAppearing();
            _viewModel.OnAppearing();
        }
        private void SearchBar_SearchButtonPressed(object sender,
            EventArgs e){
            SearchBar searchBar = (SearchBar)sender;
            string search = searchBar.Text;
            _viewModel.ExecuteLoadItemsSearchCommand(search);
        }
    }
}

```

Листинг Д.16 – Файл ItemsPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="App6.Views.ItemsPage"
  Title="{Binding Title}"
  xmlns:local="clr-namespace:App6.ViewModels"
  xmlns:model="clr-namespace:App6.Models"
  BackgroundColor="{DynamicResource PageBackgroundColor}"
  x:Name="BrowseItemsPage">
  <StackLayout>
    <SearchBar TextChanged="SearchBar_SearchButtonPressed"/>
    <RefreshView x:DataType="local:ItemsViewModel"
      Command="{Binding LoadItemsCommand}"
      IsRefreshing="{Binding IsBusy, Mode=TwoWay}">
      <CollectionView x:Name="ItemsListView"
        ItemsSource="{Binding Items}"
        SelectionMode="None">
        <CollectionView.ItemTemplate>
          <DataTemplate>
            <Grid Padding="10"
              x:DataType="model:Item"
              RowDefinitions="Auto, *"
              ColumnDefinitions="Auto, *">
              <Image Grid.RowSpan="2"
                Source="{Binding Description}"
                Aspect="AspectFit"
                BackgroundColor="White"
                VerticalOption="CenterAndExpand"
                HorizontOptions="StartAndExpand"
                HeightRequest="90"
                WidthRequest="90"/>
              <Label Grid.Column="1"
                Text="{Binding Text}"
                Padding="10"
                LineBreakMode="WordWrap"
                Margin="0,20,0,0"
                TextColor="{DynamicResource
                  TextColor}"
                VerticalOption="CenterAndExpand"
                HorizontOptions="StartAndExpand"
                Style="{DynamicResource
                  ListItemTextStyle}"
                FontSize="16"/>
            <Grid.GestureRecognizers>
              <TapGestureRecognizer
                NumberOfTapsRequired="1"
                Command="{Binding
                  Source={RelativeSource
                    AncestorType={x:Type

```

```

                                local:ItemsViewModel}},
                                Path=ItemTapped}"
                                ComandParameter="{Binding .}">
                            </TapGestureRecognizer>
                        </Grid.GestureRecognizers>
                    </Grid>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </RefreshView>
</StackLayout>
</ContentPage>

```

Листинг Д.17 – Файл ItemsViewModel.cs

```

using App6.Models;
using App6.Views;
using System;
using System.Collections.ObjectModel;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Threading.Tasks;
using Xamarin.Forms;
namespace App6.ViewModels{
    [QueryProperty(nameof(ItemId), nameof(ItemId))]
    public class ItemsViewModel : BaseViewModel{
        private Item _selectedItem;
        private int itemId;
        public ObservableCollection<Item> Items { get; }
        public ObservableCollection<Item> Items1 { get; }
        public Command LoadItemsCommand { get; }
        public Command AddItemCommand { get; }
        public Command<Item> ItemTapped { get; }
        public int ItemId{
            get{
                return itemId;
            }
            set{
                itemId = value;
            }
        }
        public ItemsViewModel(){
            Title = "Лекарства";
            Items = new ObservableCollection<Item>();
            Items1 = new ObservableCollection<Item>();
            LoadItemsCommand = new Command(() =>
                ExecuteLoadItemsCommand());
            ItemTapped = new Command<Item>(OnItemSelected);
        }
        public void ExecuteLoadItemsSearchCommand(string search){
            IsBusy = true;
            try{

```

```

Items.Clear();
Items1.Clear();
SqlConnection cn = new
    SqlConnection("workstation
id=KSENYAPHARMA.mssql.somee.com; packet
size=4096; user id=kkk_SQLLogin_2;
pwd=ydk4sfi7ym; data
source=KSENYAPHARMA.mssql.somee.com;
persist security info=False; initial
catalog=KSENYAPHARMA");
if (cn.State ==
    System.Data.ConnectionState.Closed){
    cn.Open();
    SqlCommand cmd = new
        SqlCommand("GetMedsFromCat", cn);
    cmd.CommandType =
        System.Data.CommandType.StoredProcedure;
    SqlParameter par = new SqlParameter{
        ParameterName = "@C",
        Value = itemId
    };
    cmd.Parameters.Add(par);
    SqlDataReader reader =
        cmd.ExecuteReader();
    while (reader.Read()){
        object id = reader.GetValue(0);
        object img = reader.GetValue(1);
        object name = reader.GetValue(2);
        if
            (img.ToString().ToUpper().Contains(search.ToUpper())){
            Item i = new
                Item(Int32.Parse(id.ToString()), img.ToString(),
                    name.ToString());
            Items.Add(i);
        }
    }
    cn.Close();
}
}
catch (Exception ex){
    Console.WriteLine(ex.Message);
}
finally{
    IsBusy = false;
}
}
void ExecuteLoadItemsCommand(){
    IsBusy = true;
    try{

```

```

Items.Clear();
Items1.Clear();
SqlConnection cn = new
    SqlConnection("workstation
id=KSENYAPHARMA.mssql.somee.com;packet
size=4096;user
id=kkk_SQLLogin_2;pwd=ydk4sfi7ym;data
source=KSENYAPHARMA.mssql.somee.com;persis
t security info=False;initial
catalog=KSENYAPHARMA");
if (cn.State ==
    System.Data.ConnectionState.Closed){
    cn.Open();
    SqlCommand cmd = new
        SqlCommand("GetMedsFromCat", cn);
    cmd.CommandType =
        System.Data.CommandType.StoredProcedure;
    SqlParameter par = new SqlParameter{
        ParameterName = "@C",
        Value = itemId
    };
    cmd.Parameters.Add(par);
    SqlDataReader reader =
        cmd.ExecuteReader();
    while (reader.Read()){
        object id = reader.GetValue(0);
        object img = reader.GetValue(1);
        object name = reader.GetValue(2);
        Item image = new
            Item(Int32.Parse(id.ToString()),
                img.ToString(),
                name.ToString());
        Items.Add(image);
    }
    cn.Close();
}
}
catch (Exception ex){
    Console.WriteLine(ex.Message);
}
finally{
    IsBusy = false;
}
foreach (Item i in Items){
    Items1.Add(i);
}
}
public void OnAppearing(){
    IsBusy = true;
    SelectedItem = null;
}

```

```

        public Item SelectedItem{
            get => _selectedItem;
            set{
                SetProperty(ref _selectedItem, value);
                OnItemSelected(value);
            }
        }
        async void OnItemSelected(Item item){
            if (item == null)
                return;
            await
                Shell.Current.GoToAsync($"{nameof(ItemDetailPage)}?{nameof(ItemDetailViewModel.ItemId)}={item.Id}");
        }
    }
}

```

Листинг Д.18 – Файл ItemDetailPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App6.Views.ItemDetailPage"
    xmlns:local="clr-namespace:App6.ViewModels"
    xmlns:model="clr-namespace:App6.Models"
    BackgroundColor="{DynamicResource PageBackgroundColor}"
    Title="{Binding Text}">
    <RefreshView x:DataType="local:ItemDetailViewModel"
        Command="{Binding LoadItemCommand}"
        IsRefreshing="{Binding IsBusy, Mode=TwoWay}">
        <StackLayout Spacing="25"
            Padding="10">
            <Image Source="{Binding Description}"
                HeightRequest="200"
                BackgroundColor="White"
                Aspect="AspectFit"/>
            <Label Text="{Binding Text}"
                FontSize="20"
                TextColor="{DynamicResource TextColor}"/>
            <CollectionView x:Name="ItemsDetailListView"
                ItemsSource="{Binding Pharms}"
                SelectionMode="None">
            <CollectionView.ItemTemplate>
            <DataTemplate>
            <Grid Padding="5,10,5,5"
                BackgroundColor="{DynamicResource ListBackgroundColor}"
                x:DataType="model:Pharm"
                RowDefinitions="Auto, *"
                ColumnDefinitions="Auto, *">
            <Label Grid.Column="0"

```

```

        Grid.ColumnSpan="2"
        Text="{Binding Name}"
        TextColor="{DynamicResource TextColor}"
        LineBreakMode="NoWrap"
        VerticalOptions="CenterAndExpand"
        Style="{DynamicResource
            ListItemTextStyle}"
        FontSize="13"/>
<Label Grid.Column="2"
        Text="{Binding Cost}"
        TextColor="{DynamicResource TextColor}"
        LineBreakMode="NoWrap"
        Style="{DynamicResource
            ListItemTextStyle}"
        VerticalOptions="CenterAndExpand"
        HorizontalOptions="End"
        FontSize="12"/>
<Button Grid.Column="3"
        Text="Заказать"
        FontSize="11"
        BackgroundColor="{DynamicResource
            NavigationBarColor}"
        WidthRequest="100"
        HeightRequest="33"
        Command="{ Binding
            BindingContext.ItemTapped,
            Source={x:Reference
                Name=ItemsDetailListView}}"
        CommandParameter="{Binding .}"
        VerticalOptions="Center"
        HorizontalOptions="End"/>
</Grid>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</StackLayout>
</RefreshView>
</ContentPage>

```

Листинг Д.19 – Файл ItemDetailViewModel.cs

```

using App6.Models;
using System;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Threading.Tasks;
using Xamarin.Forms;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using App6.Views;
using System.Windows.Input;
namespace App6.ViewModels{

```

```

[QueryProperty(nameof(ItemId), nameof(ItemId))]
public class ItemDetailViewModel : BaseViewModel{
    private int itemId;
    public string text;
    public string description;
    public List<Pharm> phs;
    public ObservableCollection<Pharm> Pharms { get; }
    public Command LoadItemCommand { get; }
    public Command<Pharm> ItemTapped { get; }
    public int Id { get; set; }
    public string Text{
        get => text;
        set => SetProperty(ref text, value);
    }
    public string Description{
        get => description;
        set => SetProperty(ref description, value);
    }
    public int ItemId{
        get
        {
            return itemId;
        }
        set
        {
            itemId = value;
            LoadItemId(value);
        }
    }
    public ItemDetailViewModel(){
        Pharms = new ObservableCollection<Pharm>();
        LoadItemCommand = new Command(() =>
            LoadItemId(itemId));
        ItemTapped = new Command<Pharm>(OnItemSelected);
    }
    public void LoadItemId(int itemId){
        phs = new List<Pharm>();
        Pharms.Clear();
        SqlConnection cn = new SqlConnection("workstation
            id=KSENYAPHARMA.mssql.somee.com;packet
            size=4096;user
            id=kkk_SQLLogin_2;pwd=ydk4sfi7ym;data
            source=KSENYAPHARMA.mssql.somee.com;persist
            security info=False;initial
            catalog=KSENYAPHARMA");
        try{
            if (cn.State ==
                System.Data.ConnectionState.Closed){
                cn.Open();
                SqlCommand cmd = new
                    SqlCommand("GetMedInfo", cn);

```

```

        cmd.CommandType =
            System.Data.CommandType.StoredProcedure;
        SqlParameter par = new SqlParameter{
            ParameterName = "@I",
            Value = itemId
        };
        cmd.Parameters.Add(par);
        SqlDataReader reader =
            cmd.ExecuteReader();
        while (reader.Read()){
            object name = reader.GetValue(0);
            object img = reader.GetValue(1);
            object pha = reader.GetValue(2);
            object cost = reader.GetValue(3);
            object o = reader.GetValue(4);
            Id = itemId;
            Text = name.ToString();
            Description = img.ToString();
            string cs = cost.ToString();
            cs += " руб";
            Pharm p = new
                Pharm(Int32.Parse(o.ToString()),
                    pha.ToString(), cs);
            Pharms.Add(p);
        }
        cn.Close();
    }
}
catch (Exception ex){
    Console.WriteLine(ex.Message);
}
}
async void OnItemSelected(Pharm item){
    if (item == null)
        return;
    await
        Shell.Current.GoToAsync($"{nameof(OrderPage)}?{
            nameof(OrderViewModel.ItemId)}={item.Id}");
}
}
}

```

Листинг Д.20 – Файл OrderPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App6.Views.OrderPage"
    Title="{Binding Text}"
    xmlns:local="clr-namespace:App6.ViewModels"
    xmlns:model="clr-namespace:App6.Models"

```

```

BackgroundColor="{DynamicResource PageBackgroundColor}"
x:Name="BrowseOrderPage">
<RefreshView x:DataType="local:OrderViewModel"
    Command="{Binding LoadItemCommand}"
    IsRefreshing="{Binding IsBusy, Mode=TwoWay}">
<StackLayout Spacing="25"
    Padding="10">
    <Label Grid.Column="0"
        Margin="5"
        Padding="5"
        Text="Выберите адрес аптеки:"
        TextColor="{DynamicResource TextColor}"
        LineBreakMode="NoWrap"
        VerticalOptions="CenterAndExpand"
        FontSize="20" />
    <CollectionView x:Name="OrderListView"
        ItemsSource="{Binding Orders}"
        SelectionMode="None">
    <CollectionView.ItemTemplate >
    <DataTemplate >
    <Grid BackgroundColor="{DynamicResource
        ListBackgroundColor}"
        x:DataType="model:Order"
        RowDefinitions="Auto, *"
        ColumnDefinitions="Auto, *">
    <Label Grid.Column="0"
        Margin="10,10,0,0"
        Padding="15"
        Text="{Binding Adr}"
        TextColor="{DynamicResource TextColor}"
        LineBreakMode="NoWrap"
        VerticalOption="CenterAndExpand"
        FontSize="16" />
    <Grid.GestureRecognizers>
    <TapGestureRecognizer NumberOfTapsRequired="1"
        Command="{Binding Source={RelativeSource
            AncestorType={x:Type
                local:OrderViewModel}}}, Path=ItemTapped}"
        CommandParameter="{Binding .}">
    </TapGestureRecognizer>
    </Grid.GestureRecognizers>
    </Grid>
    </DataTemplate>
    </CollectionView.ItemTemplate>
    </CollectionView>
    </StackLayout>
    </RefreshView>
</ContentPage>

```

Листинг Д.21 – Файл OrderViewModel.cs

```
using App6.Models;
```

```

using System;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Threading.Tasks;
using Xamarin.Forms;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using App6.Views;
namespace App6.ViewModels{
    [QueryProperty(nameof(ItemId), nameof(ItemId))]
    public class OrderViewModel : BaseViewModel{
        private int itemId;
        public string text;
        public ObservableCollection<Order> Orders { get; }
        public Command LoadItemCommand { get; }
        public Command<Order> ItemTapped { get; }
        public int Id { get; set; }
        public string Text{
            get => text;
            set => SetProperty(ref text, value);
        }
        public int ItemId{
            get{
                return itemId;
            }
            set{
                itemId = value;
                LoadItemId(value);
            }
        }
        public OrderViewModel(){
            Orders = new ObservableCollection<Order>();
            LoadItemCommand = new Command(() =>
                LoadItemId(itemId));
            ItemTapped = new Command<Order>(OnItemSelected);
        }
        public void LoadItemId(int itemId){
            Orders.Clear();
            SqlConnection cn = new SqlConnection("workstation
            id=KSENYAPHARMA.mssql.somee.com; packet
            size=4096; user id=kkk_SQLLogin_2;
            pwd=ydk4sfi7ym; data
            source=KSENYAPHARMA.mssql.somee.com; persist
            security info=False; initial
            catalog=KSENYAPHARMA");
            try{
                if (cn.State ==
                    System.Data.ConnectionState.Closed){
                    cn.Open();
                    SqlCommand cmd = new SqlCommand("FindApt",
                        cn);

```

```

        cmd.CommandType =
            System.Data.CommandType.StoredProcedure;
        SqlParameter par = new SqlParameter{
            ParameterName = "@I",
            Value = itemId
        };
        cmd.Parameters.Add(par);
        SqlDataReader reader =
            cmd.ExecuteReader();
        int i = 0;
        while (reader.Read()){
            object name = reader.GetValue(0);
            object img = reader.GetValue(1);
            Id = itemId;
            Text = name.ToString();
            i++;
            string cs = img.ToString();
            Order p = new Order(i,
                name.ToString(), cs);
            Orders.Add(p);
        }
        cn.Close();
    }
}
catch (Exception ex){
    Console.WriteLine(ex.Message);
}
}
async void OnItemSelected(Order item){
    if (item == null) return;
    await
        Shell.Current.GoToAsync($"{nameof(NotYetPage)}?
            {nameof(NotYetViewModel.ItemId)}={item.Id}");
}
}
}

```

Листинг Д.22 – Файл Category.cs

```

using System;
namespace App6.Models{
    public class Category{
        public int Id { get; set; }
        public string Text { get; set; }
        public Category(int i, string t)
        {
            Id = i;
            Text = t;
        }
    }
}

```

Листинг Д.23 – Файл Item.cs

```
using System;
namespace App6.Models{
    public class Item{
        public int Id { get; set; }
        public string Text { get; set; }
        public string Description { get; set; }
        public Item(int i, string t, string d){
            Id = i;
            Text = t;
            Description = d;
        }
    }
}
```

Листинг Д.24 – Файл Pharm.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace App6.Models{
    public class Pharm{
        public Pharm(int id, string name, string cost){
            Id = id;
            Name = name;
            Cost = cost;
        }
        public int Id { get; set; }
        public string Name { get; set; }
        public string Cost { get; set; }
    }
}
```

Листинг Д.25 – Файл User.cs

```
using System;
namespace App6.Models{
    public class User{
        public int Id { get; set; }
        public string Name { get; set; }
        public string Pwd { get; set; }
        public string Theme { get; set; }
        public int Auth { get; set; }
        public User(int i, string n, string p, string t, int a){
            Id = i;
            Name = n;
            Pwd = p;
            Theme = t;
            Auth= a;
        }
    }
}
```

```

    }
}
}

```

Листинг Д.26 – Файл BaseViewModel.cs

```

using App6.Models;
using App6.Services;
using App6.Themes;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using Xamarin.Forms;
namespace App6.ViewModels{
    public class BaseViewModel : INotifyPropertyChanged{
        public IDataStore<Item> DataStore =>
            DependencyService.Get<IDataStore<Item>>();
        bool isBusy = false;
        public bool IsBusy{
            get { return isBusy; }
            set { SetProperty(ref isBusy, value); }
        }
        string title = string.Empty;
        public string Title{
            get { return title; }
            set { SetProperty(ref title, value); }
        }
        protected bool SetProperty<T>(ref T backingStore, T
            value, [CallerMemberName] string propertyName = "",
            Action onChanged = null){
            if (EqualityComparer<T>.Default.Equals(backingStore,
                value))
                return false;
            backingStore = value;
            onChanged?.Invoke();
            OnPropertyChanged(propertyName);
            return true;
        }
        #region INotifyPropertyChanged
        public event PropertyChangedEventHandler PropertyChanged;
        protected void OnPropertyChanged([CallerMemberName]
            string propertyName = ""){
            var changed = PropertyChanged;
            if (changed == null)
                return;
            changed.Invoke(this, new
                PropertyChangedEventArgs(propertyName));
        }
        #endregion
    }
}

```

Приложение Е
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРИЛОЖЕНИЯ
(рекомендуемое)

Ксения123

Логин может содержать только цифры!
Логин должен состоять из 11 символов!

....

Пароль не может содержать недопустимые символы!
Пароль должен состоять минимум из 6 символов!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.1 – Результат тестирования регистрации при вводе логина
«Ксения123» и пароля «psw-»

Ксения12345

Логин может содержать только цифры!

....

Пароль не может содержать недопустимые символы!
Пароль должен состоять минимум из 6 символов!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.2 – Результат тестирования регистрации при вводе логина
«Ксения12345» и пароля «123-»

Ксе

Логин может содержать только цифры!
Логин должен состоять из 11 символов!

.....

Пароль не может содержать недопустимые символы!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.3 – Результат тестирования регистрации при вводе логина «Ксе» и пароля «pass.word»

Ксения12345

Логин может содержать только цифры!

.....

Пароль не может содержать недопустимые символы!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.4 – Результат тестирования регистрации при вводе логина «Ксения12345» и пароля «pass.word»

8987519

Логин должен состоять из 11 символов!

.....

Пароль не может содержать недопустимые символы!
Пароль должен состоять минимум из 6 символов!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.5 – Результат тестирования регистрации при вводе логина «8987519»
и пароля «pswd-»

89875190842

.....

Пароль не может содержать недопустимые символы!
Пароль должен состоять минимум из 6 символов!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.6 – Результат тестирования регистрации при вводе логина
«89875190842» и пароля «pswd-»

8987519

Логин должен состоять из 11 символов!

.....

Пароль не может содержать недопустимые символы!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.7 – Результат тестирования регистрации при вводе логина «8987519» и пароля «pass.word»

89875190842

.....

Пароль не может содержать недопустимые символы!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.8 – Результат тестирования регистрации при вводе логина «89875190842» и пароля «pswd.1»

к89875

Логин может содержать только цифры!
Логин должен состоять из 11 символов!

.....

Пароль должен состоять минимум из 6 символов!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.9 – Результат тестирования регистрации при вводе логина «к89875»
и пароля «pwd1»

к8987519084

Логин может содержать только цифры!

.....

Пароль должен состоять минимум из 6 символов!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.10 – Результат тестирования регистрации при вводе логина
«к8987519084» и пароля «pwd1»

к8987

Логин может содержать только цифры!
Логин должен состоять из 11 символов!

.....

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.11 – Результат тестирования регистрации при вводе логина «к8987»
и пароля «password»

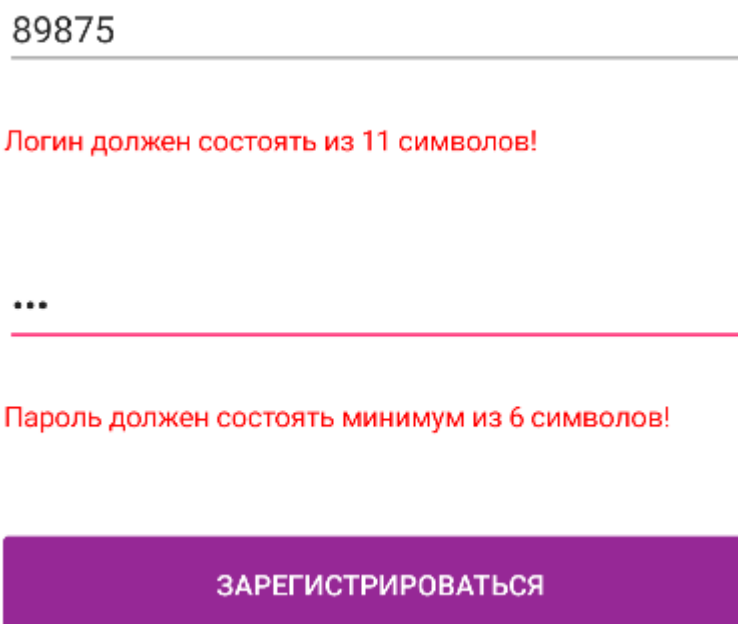
ksenya.k

Логин может содержать только цифры!
Логин должен состоять из 11 символов!

.....

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.12 – Результат тестирования регистрации при вводе логина
«ksenya.k» и пароля «password»



89875

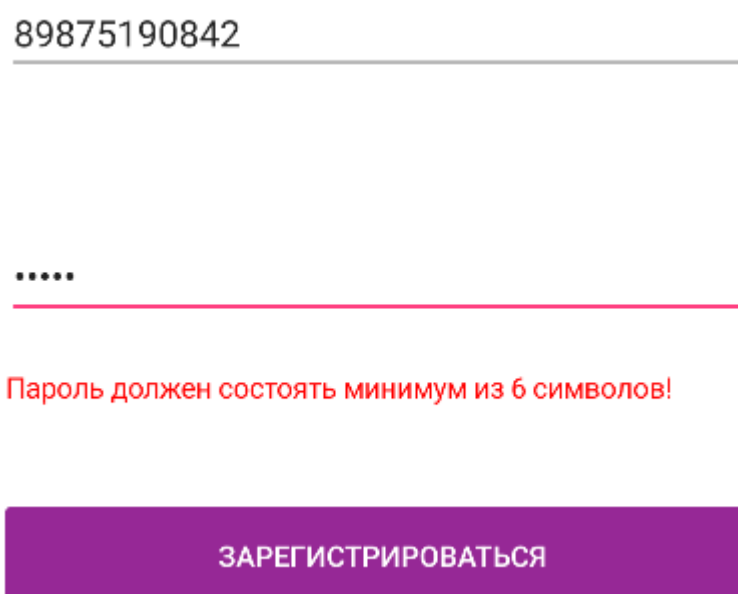
Логин должен состоять из 11 символов!

...

Пароль должен состоять минимум из 6 символов!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.13 – Результат тестирования регистрации при вводе логина «89875»
и пароля «psw»



89875190842

....

Пароль должен состоять минимум из 6 символов!

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.14 – Результат тестирования регистрации при вводе логина «Логин:
89875190842» и пароля «qwe12»

89875

Логин должен состоять из 11 символов!

.....

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.15 – Результат тестирования регистрации при вводе логина «89875» и пароля «password»

89875190840

Логин уже занят!

.....

ЗАРЕГИСТРИРОВАТЬСЯ

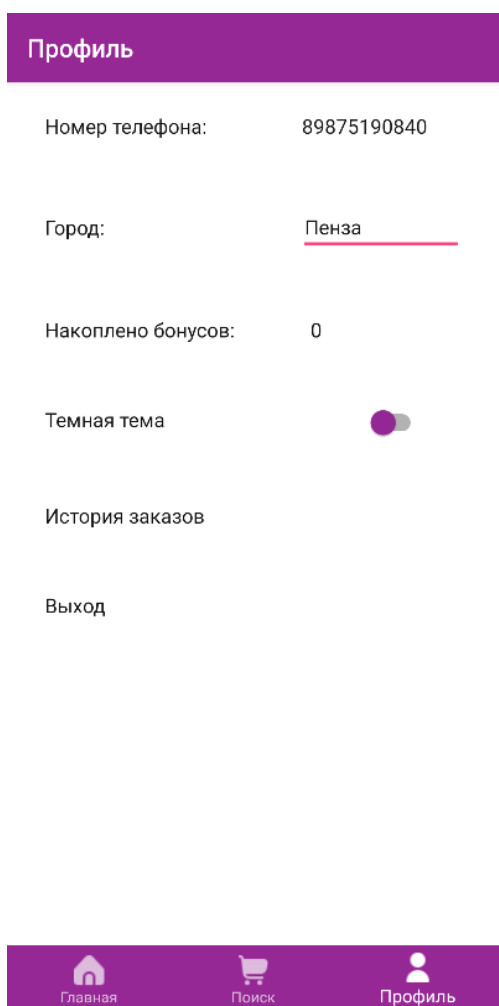
Рисунок Е.16 – Результат тестирования регистрации при вводе занятого логина

89875190842

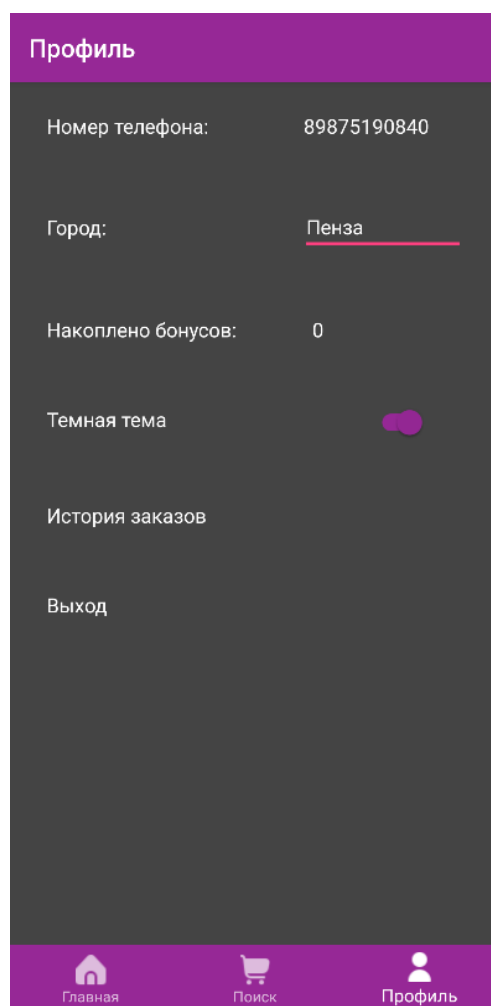
.....

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок Е.17 – Результат тестирования регистрации при вводе свободного
логина

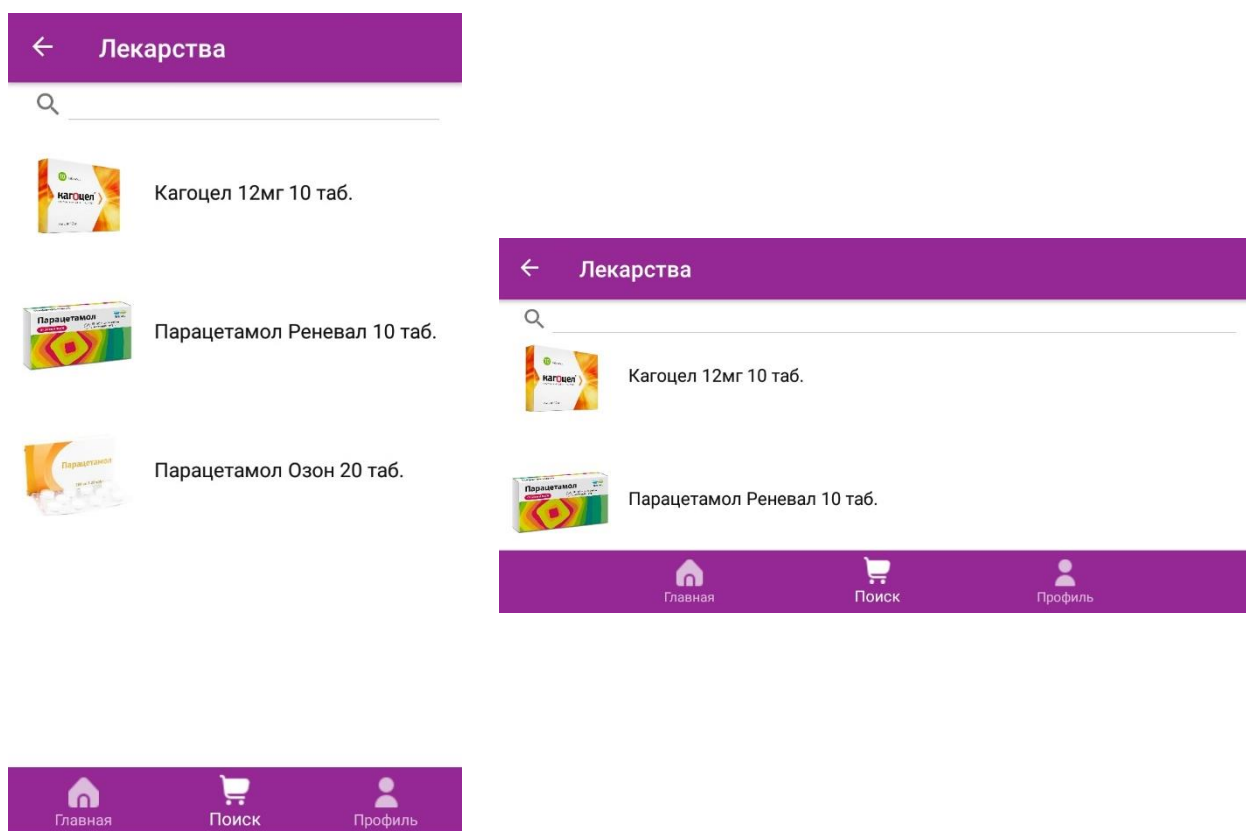


а) Светлая тема



б) Темная тема

Рисунок Е.18 – Тема интерфейса приложения



а) Книжная ориентация

б) Альбомная ориентация

Рисунок Е.19 – Ориентация экрана