

## СОДЕРЖАНИЕ

Введение .....	3
1 Общие сведения об организации ОДО «Альфа» .....	5
1.1 Структура и направления деятельности предприятия .....	5
1.2 Должностная инструкция инженера-программиста .....	5
1.3 Используемые методологии разработки программного обеспечения .....	7
1.4 Используемые методы управления жизненным циклом проекта .....	8
2 Аналитический обзор литературы .....	10
2.1 Нормативное регулирование взаимоотношений родителей и представителей школы, обеспечения деятельности органов самоуправления .....	10
2.2 Сравнительный анализ существующих программных решений и информационных систем в сфере поддержки взаимодействия представителей школы с организациями самоуправления класса и школы .....	13
2.3 Перспективы развития платежной системы и безналичных расчетов в Республике Беларусь .....	13
2.4 Виды и классификация безналичных расчетов .....	22
2.5 Архитектура программного комплекса .....	22
3 Структура программного комплекса .....	25
3.1 Структура базы данных .....	25
3.2 Структура <i>web</i> -приложения .....	26
3.3 Структура мобильного приложения .....	28
3.4 Ролевые политики для доступа к ресурсам и функциям системы .....	30
3.5 Подробный функционал и интерфейс мобильного приложения .....	31
Заключение .....	32
Список использованных источников .....	33
Приложение А Листинг программы .....	34

## ВВЕДЕНИЕ

В настоящее время на рынке существует большое количество решений для автоматизации практически любой деятельности организаций.

Говорить о том, что каждое из этих решений идеально впишется в рамки бизнес-процессов, протекающих в любой организации, не приходится. Стандартные решения автоматизации способны удовлетворить многие, но не все требования организаций. В независимости от отрасли, или сферы ведения хозяйственной деятельности одно и то же готовое средство может оказаться излишне функциональным – в лучшем случае, или иметь изъян в виду своей ограниченности.

Можно выделить ряд проблем возникающих у руководителей подразделений и ИТ специалистов, занимающихся разработкой, а также внедрением подобных систем.

Основные проблемы автоматизации:

- недостатки функциональной части систем автоматизаций;
- невозможность сопряжения функциональных модулей, систем автоматизации конкурирующих разработчиков;
- различные форматы входных и выходных данных;
- отсутствие возможностей конфигурирования многих систем автоматизации.

Тема автоматизации в настоящее время встаёт наиболее актуально. Большинство задач, предприятие вынуждено автоматизировать, так как нет возможности постоянного управления в лице человека.

Большинство задач предприятий из них являются рутинными, которые без проблем можно возложить на «плечи систем автоматизации».

Актуальность темы автоматизации так же подчёркивают относительную новизну систем с возможностью конфигурирования, возможность эффективного решения практических задач.

Преддипломная практика является неотъемлемой частью учебного процесса, в ходе которой закрепляются теоретические знания на производстве, направленной на получение практических знаний и навыков профессиональной деятельности предприятия. Преддипломная практика специализирована для расширения представлений о специальности, полученных при теоретическом обучении, а также для приобретения производственного опыта и конкретных производственных навыков по специальности.

В настоящее время существует множество областей производства в ходе функционирования, которых большая часть времени тратится на учёт различной

информации и её обработку, поэтому возникает необходимость в автоматизации таких областей. Правильно организованная автоматизация какой-либо области позволяет в разы сократить расходы предприятий и затрачиваемое время на выполнение каких-либо операций. Успешность реализации автоматизированной информационной системы напрямую зависит от выбранных средств разработки.

Местом проведения преддипломной практики было выбрано предприятие «Альфа». Основное направление деятельности: внедрение информационных технологий для бухгалтерского учета государственных учреждений.

Цели преддипломной практики: закрепление, расширение, углубление и систематизация теоретических знаний, а также разработка программного обеспечения автоматизации деятельности родительского комитета школы.

Задачи преддипломной практики:

- развитие и закрепление практических навыков выполнения анализа предметной области;
- приобретение практического опыта проектирования программных систем;
- развитие и закрепление практических навыков использования языков и инструментальных средств моделирования при проектировании системы;
- развитие и закрепление практических навыков создания программных систем с использованием современных сред разработки, поддерживающих возможность командной работы, контроля проекта и версий системы;
- развитие и закрепление практических навыков разработки документации к системе.

# 1 ОБЩИЕ СВЕДЕНИЯ ОБ ОРГАНИЗАЦИИ ОДО «Альфа»

## 1.1 Структура и направления деятельности предприятия

Компания «Альфа» работает на рынке информационных технологий уже более 10 лет. Деятельность компании началась в 2003 году с автоматизации движения товарно-материальных ценностей и взаиморасчетов со сторонними организациями. После успешного внедрения и апробации данной компоненты, появились новые заказы, разрабатывались и совершенствовались новые программы. За время работы компании было разработано и внедрено на предприятиях более сорока различных программных модулей, позволяющих автоматизировать различные службы и сферы деятельности.

Таким образом, сегодня «Альфа-Офис» – это целый комплекс готовых программных продуктов для различных служб предприятий и организаций, которые могут быть использованы как отдельные компоненты, автоматизируя работу конкретных служб и департаментов, так и во взаимосвязи друг с другом, автоматизируя работу организации в целом.

Готовые программные решения охватывают практически все основные виды деятельности: производство и диспетчерская служба, снабжение и сбыт, торговля и склад, строительство, внешнеэкономическая деятельность, транспорт, бухгалтерский учет, жилищно-коммунальное хозяйство, диспетчерская такси, кадровый учет, CRM (система управления взаимодействием с клиентами) и многое другое.

В 2006 году освоены новое направление деятельности компании – оценочная деятельность (услуги по оценке недвижимости, оборудования, транспортных средств; оценка ущерба, причиненного имуществу)

С 2009 года ОДО Альфа-Гомель является официальным представителем в г. Гомеле единственного в Беларуси разработчика антивирусного ПО – компании «ВирусБлокАда».

С 2010 года в компании можно приобрести широкий спектр лицензионных программных продуктов сторонних разработчиков (операционные системы, офисные приложения, файл-менеджеры, антивирусы и многое-многое другое) [1].

## 1.2 Должностная инструкция инженера-программиста

Программист – специалист, занимающийся непосредственной разработкой программного обеспечения для различного рода вычислительно-операционных систем. В организации ОДО «Альфа» преобладают *backend*-разработчики, а также разработчики систем автоматического тестирования.

Вне зависимости от рода деятельности каждому специалисту предоставляется рабочее место, которое включает в себя персональный компьютер и набор лицензионного программного обеспечения, необходимого для выполнения его обязанностей.

Должностные обязанности инженера-программиста не ограничиваются разработкой программ, вторая и не менее важная составляющая работы инженера-программиста – внедрение созданных программных средств и оформление необходимой технической документации.

Общие положения должностной инструкции инженера-программиста:

- инженер-программист относится к категории специалистов;
- инженер-программист назначается на должность и освобождается от нее приказом генерального директора по представлению технического директора/начальника структурного подразделения;
- инженер-программист подчиняется непосредственно техническому директору/начальнику структурного подразделения;
- на время отсутствия инженера-программиста его права и обязанности переходят к другому должностному лицу, о чем объявляется в приказе по организации;
- инженер-программист должен знать руководящие и нормативные материалы, регламентирующие методы разработки алгоритмов и программ и использования вычислительной техники при обработке информации, основные принципы структурного программирования, виды программного обеспечения, технологию автоматической обработки информации и кодирования информации, формализованные языки программирования, порядок оформления технической документации.

Инженер-программист выполняет следующие должностные обязанности:

- разрабатывает технологию решения задачи по всем этапам обработки информации;
- определяет информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля;
- выполняет работу по подготовке программ к отладке и проводит отладку;
- осуществляет запуск отлаженных программ и ввод исходных данных, определяемых условиями поставленных задач;
- проводит корректировку разработанной программы на основе анализа выходных данных;
- разрабатывает инструкции по работе с программами, оформляет необходимую техническую документацию;
- определяет возможность использования готовых программных продуктов;
- осуществляет сопровождение внедрения программ и программных средств;

- разрабатывает и внедряет системы автоматической проверки правильности программ, типовые и стандартные программные средства, составляет технологию обработки информации;

- разрабатывает и внедряет системы автоматической проверки правильности программ, типовые и стандартные программные средства, составляет технологию обработки информации.

Инженер-программист несет ответственность за:

- невыполнение или несвоевременное, халатное выполнение своих должностных обязанностей;

- несоблюдение действующих инструкций, приказов и распоряжений по сохранению коммерческой тайны и конфиденциальной информации;

- нарушение правил внутреннего трудового распорядка, трудовой дисциплины, правил техники безопасности и противопожарной безопасности.

### **1.3 Используемые методологии разработки программного обеспечения**

Бизнес компаний часто зависит от качества работы информационных систем. Ошибки в ИТ-приложениях, некорректное выполнение функций могут привести к финансовым потерям. Для снижения подобных рисков ОДО «Альфа» использует при разработке программного обеспечения методологию *Continuous Integration*.

*Continuous Integration* – практика разработки программного обеспечения, которая заключается в постоянном слиянии рабочих копий в общую основную ветвь разработки (до нескольких раз в день) и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем. Это полностью автоматизированная практика создания, сборки и тестирования программного кода. При ее применении компиляция и компоновка кода начинает проводиться как можно раньше и повторяется как можно чаще. Специальное программное обеспечение отслеживает процесс разработки: при наличии изменений в коде автоматически запускается процесс сборки и тестирования. Это позволяет найти дефекты и противоречия в компонентах системы еще на ранних стадиях ее создания. В результате – обеспечивается высокое качество программного обеспечения.

Преимущества *Continuous Integration*:

- сокращение ручных операций – этапы создания, сборки и тестирования программного обеспечения проводятся в автоматическом режиме;

- наличие рабочей ИТ-системы на всем протяжении процесса разработки

- у проектной команды всегда есть свежая версия решения для демонстрации заказчику, получения обратной связи и быстрой доработки;
- качество программного обеспечения – в рамках *Continuous Integration* используются различные программные средства для контроля качества кода, что позволяет сократить количество ошибок;
- минимизация рисков – дефекты выявляются на ранних стадиях разработки информационной системы, что помогает избежать увеличения сроков и стоимости проекта;
- окупаемость инвестиций в ИТ – автоматизация процесса разработки обеспечивают высокую эффективность и надежность информационной системы.

Для разработки программного обеспечения в рамках *Continuous Integration* в ОДО «Альфа» используются различные программные и аппаратные средства. Выбор конкретных инструментов зависит от задач и специфики проекта. На всех компьютерах установлены последние версии операционной системы и другого программного обеспечения. Это требование заложено в политику безопасности компании.

#### **1.4 Используемые методы управления жизненным циклом проекта**

Начало нового проекта как правило сопровождается решением массы организационных вопросов: как будут взаимодействовать участники проекта, где будут храниться документы и как будет построено их согласование, как будут ставить задачи и выдавать поручения. Для решения этой задачи ОДО «Альфа» использует *Confluence* и *Jira*.

*Confluence* – это *wiki*-среда, в которой можно создавать, упорядочивать и обсуждаете рабочие процессы всей командой. Данная среда позволяет хранить документы с сохранением версии. По умолчанию пользователь берет всегда последнюю версию, что снижает количество ошибок. В любой момент можно вернуться к любой из предыдущих версий. Также сохраняются версии страниц, всегда можно увидеть, кто и какие изменения внес, сравнивая любые две версии попарно.

*Jira* также является разработкой *Atlassian* и создана как система регистрации и исполнения запросов на обслуживание. Все текущие задания хранятся в виде списка с возможностью сортировки и фильтрации. При этом заданиям можно присваивать статусы и ярлыки.

Для управления жизненным циклом разрабатываемых проектов в компании ОДО «Альфа» используется системы контроля версия *Git*.

*Git* – распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра *Linux*, первая версия выпущена 7 апреля 2005 года. Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе *Git* или пользовательские интерфейсы. Например, *Cogito* является именно таким примером оболочки к репозиториям *Git*, а *StGit* использует *Git* для управления коллекцией исправлений.

Репозиторий *Git* представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого существующих в хранилище файлов.

Удалённый доступ к репозиториям *Git* обеспечивается *git-daemon*, *SSH*-или *HTTP*-сервером. *TCP*-сервис *git-daemon* входит в дистрибутив *Git* и является наряду с *SSH* наиболее распространённым и надёжным методом доступа. Метод доступа по *HTTP*, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.



## **2 АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ**

### **2.1 Нормативное регулирование взаимоотношений родителей и представителей школы, обеспечения деятельности органов самоуправления**

Нормативное регулирование - особая форма деятельности людей в обществе, направленная на создание, реализацию и обеспечение различного рода правил, т.е. социальных норм поведения людей, с целью упорядочения их деятельности и достижения стабильности в масштабах всего общества.

Признаки нормативного регулирования:

- системность: общество использует множество социальных регуляторов (право, мораль, идеологию, религию), объединив их в систему, в которой каждый регулятор занимает своё, специально отведенное для него место;
- нормативность: регулирование на основе социальных норм, рассчитанное на общее массовое применение в типичной обстановке;
- объективность: нормативное регулирование объективно обусловлено общественными потребностями;
- конкретно-исторический характер: выражается в том, что система нормативного регулирования характерна для каждого конкретно-исторического общества;
- регулятивность: сущность нормативного регулирования в подчинении людей объединенным общим стандартом, заключении поведения людей в определенные рамки целевой характер;
- нормативное регулирование направлено на достижение общественного порядка социальный характер;
- нормативное регулирование направлено на упорядочение общения людей друг с другом.

В данном случае необходимо отличать социальное нормативное регулирование от технического, устанавливающего правила и стандарты общения людей с техникой обеспеченность мерами социального принуждения. Каждый социальный регулятор содержит специальное средство, направленное на обеспечение его эффективности, то есть меры принуждения.

Механизм нормативного регулирования – это процесс формирования общественного порядка на основании социальных норм, объединяющее в себе стадии и средства нормативного регулирования.

Элементы:

- социальные потребности;
- социальные нормы;

- реализация социальных норм;
- общественные отношения;
- меры социального принуждения.

Стадии:

- формирование социальных норм на основе общественных потребностей;
- реализация социальных норм;
- формирование новых общественных отношений;
- включение новых общественных отношений в общественный порядок, его развитие;
- дополнительная стадия применения мер социального принуждения.

Самоуправление в учреждении образования – это взаимодействие учащихся и их родителей, педагогов, администрации и общественности, регулируемое нормативными актами и осуществляемое специальными органами самоуправления.

Деятельность школьного самоуправления направлена на:

- защиту прав и интересов учащихся;
- помощь учащимся в реализации творческих способностей, в поиске своего места в обществе, в утверждении своей жизненной позиции;
- пропаганду здорового образа жизни;
- поддержку общественно ценных инициатив школьников.

Цель родительского самоуправления – активное участие в управлении школой и своей ассоциацией, оказание помощи школе в достижении высокого качества воспитания и обучения детей.

Высшим органом самоуправления родителей является родительское собрание, созываемое по необходимости, но не реже двух раз в год, а в период между родительскими собраниями – школьный родительский комитет, избранный собранием сроком на 2 года.

Школьный родительский комитет:

- готовит и проводит родительские собрания и другие мероприятия родителей;
- организует выполнение решений, принятых школьным родительским собранием;
- изучает общественное мнение и потребности родителей;
- планирует свою работу и организует выполнение намеченных планов;
- создает родительский финансовый фонд и принимает решения об его использовании в интересах совершенствования деятельности школы, оздоровления, обучения и воспитания учащихся;
- формирует постоянные и временные комиссии (или иные рабочие органы) по отдельным направлениям деятельности.

Высшим органом родительского самоуправления в классе является их общее собрание, созываемое по мере необходимости, но не реже раза в учебную четверть. В период между родительскими собраниями работу родителей организует классный родительский комитет, избираемый сроком на 2 года.

Классный родительский комитет:

- готовит и проводит классные родительские собрания и другие родительские мероприятия;
- организует выполнение решений родительского собрания и вышестоящих органов самоуправления родителей;
- планирует свою работу и обеспечивает выполнение намеченных планов;
- создает при необходимости постоянные и временные родительские комиссии и иные органы для решения отдельных вопросов;
- выясняет мнение родителей по актуальным проблемам жизни школы и класса, по обсуждаемым в школе проектам нормативных документов, изучает потребности и интересы родителей в школе;
- обеспечивает участие родителей во внеклассной и внешкольной воспитательной работе с детьми, в ремонте классного помещения, в улучшении условий жизни и учебы детей в школе и классе;
- организует обмен опытом семейного воспитания, поощряет родителей, хорошо воспитывающих своих детей.

Свои заседания родительский комитет класса проводит по мере необходимости, но не реже одного раза в учебную четверть. Родительский комитет класса организует свою работу в тесном взаимодействии с классным руководителем и классным ученическим советом.

Органы самоуправления родителей должны руководствоваться следующими принципами:

- каждый орган самоуправления наделяется реальными властными полномочиями;
- полномочия, принимаемые решения и осуществляемые действия не должны ущемлять интересы и права других участников школьной жизни;
- все школьные объединения и коллективы должны взаимодействовать на основе принципов совета, согласия, сотрудничества и соуправления;
- в школе действует принятая в соответствии с ее Уставом согласительная система разрешения конфликтов и противоречий между органами самоуправления;
- между всеми органами самоуправления осуществляется координация и взаимодействие по актуальным школьным проблемам;

– в своей деятельности органы школьного самоуправления руководствуются принципами законности, педагогической целенаправленности, коллегиальности в принятии решений и персональной ответственности за них;

Работа любого органа родительского самоуправления в школе начинается с выявления актуальных потребностей и интересов своих родителей, на основе которых затем определяются содержание и основные направления его деятельности и организационная структура.

## **2.2 Сравнительный анализ существующих программных решений и информационных систем в сфере поддержки взаимодействия представителей школы с организациями самоуправления класса и школы.**

На сегодняшний день существует огромное количество аналогичных программных продуктов. Часть аналогов не содержит необходимый функционал, а часть программ содержит набор излишних функций, из-за которых цена на программный продукт растет, однако не все могут это себе позволить.

Среди существующих программных решений можно выделить мобильное приложение «Родком», которое разрабатывается специально для сбора денег на нужды класса и оплаты различных школьных мероприятий. Недостатками данного приложения является наличие большого количества рекламы в бесплатной версии приложения, а также возможность использования только для обучения в садах и школах Санкт-Петербурга.

В качестве альтернативы родители используют различные мессенджеры и чаты, однако данный подход подходит лишь для коммуникации, а средства по-прежнему собираются на картах различных банков, либо же наличными, что в наше время является очень неудобным способом.

Различные компании предлагают свои решения данной проблемы. Одной из таких компаний является Яндекс, которая предлагает создать кошелек для сбора средств. По ссылке можно пополнять данный кошелек с карты или других электронных кошельков. Для конкретного сбора к ссылке можно подставить фиксированную сумму – всем, кто платит, не придётся переспрашивать и вводить её вручную.

Минусом данного способа является отсутствие у старших поколений необходимых знаний в интернет-сети для проведения подобных операций, кроме того, очень сложно отследить в автоматическом режиме кто произвел оплату из родителей, а кто нет.

Для личных сборов существует площадка *GoFundMe*. Люди используют *GoFundMe* для того, чтобы собрать деньги на свадьбы, похороны и все, что происходит между этими событиями.

В русских источниках пишут, что платформа общемировая, и можно начать сбор из любой точки мира. Жители США могут получить свои деньги через платежную систему *WePay*. Все остальные работают через *PayPal*.

Данное приложение лишь отдаленно напоминает то, что нужно заказчику. Обычным родителям в наше время очень тяжело работать с иностранными электронными кошельками, к тому же, для жителей нашей страны есть ограничения по использованию. Существуют риски, что электронный кошелек будет заморожен и для возврата средств необходимо будет разбираться с администрацией сайта. Исходя из этого данное решение не является хорошим для целей, необходимых заказчику.

Еще одним способом собирать средства на какие-либо мероприятия является программное решение от социальной сети Вконтакте – «Цели». Он доступен администратору любого сообщества в социальной сети.

Для доступа к «Целям» необходимо скачать приложение из маркета Вконтакте. Функционал «Целей» напоминает возможности других краудфандинговых платформ.

Пользователи могут помочь донату переводом с мобильного телефона, банковской карты или электронного кошелька в системе «Яндекс.Деньги». Переводы от юридических лиц принимаются через платежную систему *CloudPayments*.

Представители социальной сети обещают, что деньги будут зачисляться сразу и организаторам не придется ждать окончания срока кампании. Скорость поступления средств на счет зависит только от платежных систем.

Приложение «Цели» самое лучшее по мнению заказчика, однако и здесь есть свои недостатки: для создания нового мероприятия необходимо создавать новое сообщество в социальной сети, кроме того транзакции между картами и электронными кошельками забирает процент от переводимой суммы – этот факт является большим недостатком.

В таблице 1.1 сравнительный анализ плюсов и минусов рассмотренных аналогов с разрабатываемым программным продуктом. Главным преимуществом разработанного программного продукта является всего функционала, необходимого заказчику, а также бесплатное распространение продукта, кроме того все собрано в одном мобильном приложении, что является очень удобным для родителей.

Таблица 1.1 – Сводная таблица сравнения аналогов и программного средства курсового проекта

	Родком	Мессенджеры + Яндекс.Деньги	<i>GoFundMe</i>	Цели
Наличный расчет	нет	да	нет	нет
Безналичный расчет	да	да	да	да
Удобство в использо- вании	да	нет	нет	нет
Использование в Бе- ларуси	нет	да	нет	да
Бесплатная лицензия	да	да	да	да

Исходя из сравнительного анализа существующих программных решений можно сделать вывод, что разрабатываемое приложение не только не уступает аналогам, но и превосходит их по ряду важных параметров. Кроме того, приложение разрабатывается в согласовании с заказчиком, следовательно заказчик получит тот функционал, который он хочет в полном объеме, без всяких излишеств.

### 2.3 Перспективы развития платежной системы и безналичных расчетов в Республике Беларусь

Положение о безналичных расчетах в Республике Беларусь было утверждено постановлением Правления Национального банка Республики Беларусь от 29 марта 2001 г. (протокол № 66). После этого оно неоднократно изменялось и дополнялось. Положение о безналичных расчетах традиционно является в Республике Беларусь одним из самых нестабильных нормативных актов.

Развитие безналичных расчетов является важнейшим элементом регулирования денежно-кредитной сферы, способствующим повышению эффективности монетарной политики.

По мере сокращения доли наличных денег в структуре денежного предложения снижается общая потребность в эмиссии Национального банка для удовлетворения спроса экономики на деньги, расширяются возможности банков по управлению собственной ликвидностью, уменьшается необходимость в поддержке банков Национальным банком. В целом это ведет к усилению влияния монетарных мер на экономические процессы и обеспечению ценовой стабильности.

В то же время доля безналичных расчетов в общем объеме расчетов по розничным платежам недостаточно высока. Для ее увеличения разработан и реализу-

ется план совместных действий государственных органов и участников финансового рынка по развитию в Республике Беларусь системы безналичных расчетов по розничным платежам с использованием современных электронных платежных инструментов и средств платежа на 2013—2015 годы, утвержденный постановлением Совета Министров Республики Беларусь и Национального банка Республики Беларусь от 1 апреля 2013 г. № 246/4.

На основе анализа итогов выполнения плана в 2013 году и деятельности банковской системы по развитию системы безналичных расчетов по розничным платежам Национальным банком был разработан и представлен Президенту проект Указа, который помимо введения предельной суммы платежа наличными денежными средствами при проведении физическими лицами расчетов по оплате товаров (работ, услуг), реализуемых юридическими лицами и индивидуальными предпринимателями, устанавливает иные нормы, стимулирующие развитие безналичных расчетов, осуществляемых физическими лицами.

При подготовке Указа принят во внимание опыт зарубежных стран по ограничению расчетов наличными денежными средствами, в том числе Российской Федерации, Украины, ряда стран Европейского союза.

Указом установлен предельный размер одного платежа наличными денежными средствами при проведении физическими лицами расчетов, не связанных с осуществлением предпринимательской деятельности, по оплате товаров (работ, услуг), реализуемых юридическими лицами и индивидуальными предпринимателями, в размере 1000 базовых величин (подпункт 1.1 пункта 1 Указа).

Установленное ограничение не распространяется на платежи физических лиц за товары (работы, услуги), осуществляемые путем внесения наличных денег непосредственно в кассы банков для зачисления на банковские счета юридических лиц, индивидуальных предпринимателей (подпункт 1.2 пункта 1 Указа).

В целях создания благоприятных условий для преимущественного осуществления населением расчетов в безналичной форме до 1 января 2017 года не признаются объектом налогообложения подоходным налогом с физических лиц доходы, выплачиваемые банками, находящимися на территории Республики Беларусь, физическим лицам в виде процентов от суммы оплаты товаров (работ, услуг) с использованием банковских платежных карточек, систем дистанционного банковского обслуживания, но не более 2 % от суммы такой оплаты (подпункт 1.4 пункта 1 Указа).

Таким образом, предложенный в Указе вариант стимулирования физических лиц в размере 2 % от суммы операций при существующих объемах безналичных платежей не превышает размеров, определенных нормами Налогового кодекса Республики Беларусь, но упрощает систему учета этих средств, что будет способство-

вать активному проведению банками широкомасштабных рекламных акций, программ лояльности, бонусных программ, направленных на увеличение доли безналичных расчетов по розничным платежам. Реализация данной нормы позволит существенно расширить круг банков — участников указанных мероприятий и станет действенным фактором повышения заинтересованности населения в осуществлении безналичных платежей за товары (работы, услуги).

В Указе установлена норма, согласно которой не признаются объектом обложения подоходным налогом с физических лиц суммы денежных средств, похищенные со счетов физических лиц, к которым выданы банковские платежные карточки, и в последующем возмещенные им банками (подпункт 1.5 пункта 1 Указа). Таким образом, физические лица, ставшие жертвами мошенников в области операций с банковскими платежными карточками, при возврате банками похищенных денежных средств освобождаются от уплаты подоходного налога с физических лиц, так как, по сути, возвращаемые денежные средства не являются доходом [2].

С целью повышения эффективности, надежности и безопасности функционирования национальной платежной системы, для своевременного и качественного осуществления расчетов Национальному банку РБ необходимо продолжать работу по совершенствованию нормативно-правовой базы платежной системы.

Представляется, что направлениями совершенствования безналичных расчетов на территории Республики Беларусь являются, во-первых, приближение механизма работы платежной системы к международным требованиям и стандартам.

В соответствии с основными международными тенденциями в организации и развитии платежных систем, в Беларуси межбанковские платежи разделены на два потока: крупные и срочные платежи для обработки в системе *BISS* и на прочие (несрочные) платежи, проводимые в клиринговой системе. В системе *BISS* расчеты осуществляются в реальном режиме времени, что дает возможность банкам и их клиентам использовать денежные средства, поступившие на их счета с момента зачисления, позволяет ускорить оборачиваемость денежных средств, а также минимизировать риски в платежной системе. В клиринговой системе прочих (несрочных) платежей расчеты осуществляются путем взаимного зачета требований и обязательств участников, что позволяет сократить объемы денежных средств банков, необходимых для завершения расчетов, а также способствует сокращению затрат на обработку информации. Согласно постановлению, денежные переводы, равные или большие 3 миллиона рублей, к обработке клиринговой системе прочих платежей не принимаются, а значит, подлежат обработке в системе *BISS* и могут рассматриваться в качестве крупных денежных переводов, передаваемых в систему *BISS* со статусом «Срочный» либо «Несрочный».



Во-первых, для повышения эффективности функционирования системы межбанковских расчетов необходимо переориентировать основной поток платежей в более безопасную и надежную систему *BISS*.

Во-вторых, необходимо сокращать наличный денежный оборот. Важное место в проводимой работе по совершенствованию системы безналичных расчетов занимают выработка и осуществление мер по развитию системы безналичных расчетов физических лиц за товары (работы, услуги). Речь идет о розничных платежах. Стратегическим направлением здесь является расширение использования банковских пластиковых карточек различных модификаций как наиболее перспективного платежного инструмента. Использование пластиковых карт для безналичных расчетов имеет большие преимущества перед наличными деньгами. Для владельцев карт это оперативность расчетов; отсутствие риска потери, ограбления и ошибок в расчетах, связанных с использованием наличных денег; возможность получения процентов на остаток средств, хранящихся на картах; обеспечение конфиденциальности информации, хранящейся на карте. Для предприятий торговли: простота и оперативность обслуживания клиентов; снижение риска ограбления и сложностей, связанных с инкассацией наличных денег; оперативность перевода денежных средств на счета магазинов после инкассации. Для банка-эмитента: появление новых источников доходов за счет средств, привлеченных на карты; получение комиссионных, взимаемых с операций по картам; увеличение числа клиентов за счет предоставления услуг нового типа; уменьшение расходов на обслуживание наличного оборота. В рамках реализации этой задачи разработана Программа поэтапного внедрения системы безналичных расчетов с использованием банковских пластиковых карточек на территории Беларуси, которая 31 января 2000 года была одобрена совместным постановлением Совета Министров Республики Беларусь и Национального банка Республики Беларусь № 126/3 [3].

В основу Программы положены мировой опыт продвижения карточных продуктов, широкомасштабное внедрение технологии массового обслуживания населения по заработной плате через банки с использованием банковских пластиковых карточек (зарплатная технология).

Для внедрения зарплатной технологии должны быть созданы условия для беспрепятственного получения гражданами наличных денег через банкоматы (пункты выдачи наличных) в любое время суток. Поскольку не все деньги будут сниматься сразу, о чем свидетельствует опыт реализации пилот-проекта в Солигорске, банки получают дополнительные ресурсы, которые смогут направлять в экономику. Одновременно будет решена и психологическая проблема, связанная с недоверием населения к банку и банковской пластиковой карточке. Реализации этих задач требует создания широкой сети банкоматов.

Для выполнения карточкой функции безналичных расчетов параллельно должна проводиться работа по созданию обширной инфраструктуры на предприятиях торговли и сервиса (платежные терминалы, сети телекоммуникаций). Должны быть обеспечены, как минимум, синхронность объемов эмиссии карточек и развития инфраструктуры их использования, а еще лучше – опережающие темпы создания инфраструктуры.

В-третьих, большую роль играет развитие платежной системы. Основными задачами развития платежной системы являются:

- повышение экономической и эксплуатационной эффективности системы межбанковских расчетов при минимизации всевозможных рисков, приведение ее в соответствие с Ключевыми принципами, разработанными для системно значимых платежных систем Комитетом по платежным и расчетным системам Банка международных расчетов (г. Базель, Швейцария);
- совершенствование нормативно-правовой базы безналичных расчетов;
- внедрение в платежный оборот современных технологий платежа с использованием электронных платежных инструментов;
- достижение поставленных задач будет обеспечиваться посредством;
- оптимизации денежного обращения в направлении повышения доли безналичных расчетов во внутреннем платежном обороте;
- совершенствования системы международных расчетов банков для реализации внешнеэкономических задач страны;
- дальнейшего развития инфраструктуры платежной системы с учетом международных тенденций;
- унификация базовых характеристик национальной платежной системы с зарубежными;
- для повышения экономической и эксплуатационной эффективности национальной платежной системы, решения задач ее развития необходимо осуществление следующих основных мероприятий;
- усиление роли Национального банка в качестве органа надзора над платежной системой;
- повышение производительности и пропускной способности технических компонентов платежной системы. Расширение сферы применения электронного документооборота в платежной системе и постепенный переход на безбумажную технологию при осуществлении безналичных расчетов;
- увеличение продолжительности рабочего времени системы *BISS* и клиринговой системы с последующим переводом на круглосуточную работу;

– создание на базе системы *BISS* модернизированной системы *BISS*, сочетающей систему, построенную на валовой основе, с элементами неттинговой системы расчетов;

– повышение операционной и производственной эффективности внутрибанковских комплексов расчетных центров, и прежде всего ускорение расчетов с использованием различных платежных инструментов в пределах одного банка;

– использование Национальным банком права перераспределения денежных потоков между системой валовых расчетов и нетто-расчетов в целях минимизации рисков;

– совершенствование механизмов контроля Национальным банком за рисками национальной платежной системы;

– совершенствование механизма регулирования Национальным банком текущей ликвидности банковской системы на основе внедрения современных залоговых операций с ценными бумагами;

– улучшение качества управления Автоматизированной системой межбанковских расчетов (АС МБР), повышение эффективности использования инструментов поддержания ликвидности на уровне, необходимом для осуществления расчетов, совершенствование контроля за рисками в системе межбанковских расчетов;

– повышение уровня надежности технических комплексов и безопасности функционирования платежной системы.

В-четвертых, рост мирового платежного оборота и обусловленный этим рост издержек обращения диктуют необходимость создания принципиально нового механизма безналичного и наличного денежного обращения, обеспечивающего быстрорастущие потребности в платежах, а также ускорение оборачиваемости денежных средств при одновременном снижении издержек обращения и сокращении трудовых затрат. Возможный и основной путь решения данной проблемы это использование так называемой безбумажной технологии на основе передового опыта индустриально развитых стран в сфере применения на практике заменителей наличных денег, платежных инструментов и средств, создания технологий и технических устройств для их автоматической обработки. В условиях нарастающей конкуренции банки должны внедрять новые виды обслуживания и повышать качество традиционных услуг [4].

В таких условиях исключительно важным становится скорейшее освоение банками новой сферы деятельности – Интернет-банкинга.

Обычно под Интернет-банкингом (электронным банкингом) понимают оказание услуг банками по дистанционному (удаленному) обслуживанию через Интернет, позволяющее клиенту получать банковские услуги, не посещая при этом офис банка.

Поначалу Интернет банкинг предусматривал лишь ознакомление с информацией о самом банке, об осуществляемых им операциях и их условиях, а также о получении выписки по счету, что по существу являлось просто справочной системой.

Затем появилась возможность управления счетом. Удаленное управление счетом через Интернет обычно подразумевает проверку состояния счета, оплату разнообразных счетов и перевод средств с одного счета на другой, а также предоставление клиенту информационной поддержки и многочисленных сопутствующих услуг.

Классический вариант системы Интернет-банкинга включает в себя полный набор банковских услуг, предоставляемых клиентам – физическим и юридическим лицам в офисах банка, естественно, за исключением операций с наличными деньгами.

Как правило, с помощью систем Интернет-банкинга можно покупать и продавать безналичную валюту, оплачивать коммунальные услуги, платить за доступ в Интернет, оплачивать счета операторов сотовой и пейджинговой связи, проводить безналичные внутри- и межбанковские платежи, переводить средства по своим счетам, и, конечно, отслеживать все банковские операции по своим счетам за любой промежуток времени.

Использование системы Интернет-банкинга дает ряд преимуществ: во-первых, существенно экономится время за счет исключения необходимости посещать банк лично; во-вторых, клиент имеет возможность 24 часа в сутки контролировать собственные счета и, в соответствии с изменившейся ситуацией на финансовых рынках, мгновенно отреагировать на эти изменения (например, закрыв вклады в банке, купив или продав валюту и т.п.). Кроме того, системы Интернет-банкинга незаменимы для отслеживания операций с платежными картами – любое списание средств с карточного счета оперативно отражается в выписках по счетам,готавливаемых системами, что так же способствует повышению контроля со стороны клиента за своими операциями.

Развитие Интернет-банкинга в Беларуси сегодня зависит от ряда факторов. Можно выделить негативно и позитивно влияющие на этот процесс факторы. К негативно влияющим факторам относятся:

- значительная инертность в банковской среде по отношению к развитию Интернет-технологий, отсутствие понимания стратегического значения этих вопросов;
- неприспособленность для электронной коммерции отечественной банковской системы на технологическом уровне;
- недостаточная развитость законодательной базы для шифрования в компьютерных сетях;

- сложность налоговой системы и неприспособленность ее к электронной коммерции;
- высокая стоимость услуг Интернет-провайдеров как относительно средних доходов населения, так и в сравнении с международными ценами на Интернет-услуги;
- низкое качество каналов связи, их ненадежность и невысокая пропускная способность.

С другой стороны, существуют и позитивно влияющие факторы на развитие Интернет банкинга в Беларуси, а именно:

- среди стран СНГ Республика Беларусь занимает первое место по уровню развития телекоммуникационных сетей;
- наличие высококвалифицированных кадров, способных реализовать на практике системы, поддерживающие технологию Интернет банкинга;
- имеет место опережающее Интернет банкинг развитие электронной коммерции, хоть и не очень существенное по масштабам, но на достаточно высоком технологическом уровне;
- высокая концентрация вкладов населения всего в нескольких банках-монополистах может стимулировать развитие Интернет банкинга с целью существенного снижения издержек на эти операции.

Исходя из этих данных можно сделать следующий вывод, что банки должны активно внедрять новые виды банковских услуг, использовать современные стратегии, основанные на лучших мировых достижениях, что может обеспечить эффективный доступ к полным, отражающим состояние в реальном времени данным пользователя по нескольким каналам распределения, а также предоставит им новые возможности для развития. Для клиентов новые предложения создадут дополнительные удобства, доступ ко всем каналам (включая беспроводной) и усовершенствованные инструменты позволят осуществлять все интересующие их финансовые операции в одной «точке». Банки, которые на более высоком уровне смогут предложить финансовые услуги, окажутся лидерами в сфере электронных услуг.

## **2.4 Виды и классификация безналичных расчетов**

Безналичные расчеты совершаются с использованием различных форм. Форма безналичных расчетов представляет собой совокупность способов платежа, расчетных документов и определенного документооборота.

Безналичные расчеты в Республике Беларусь представлены следующими группами:

- банковский перевод;
- документарные операции;
- денежные переводы;
- электронные деньги.

В свою очередь, банковский перевод включает в себя такие элементы как платежное требование, платежное поручение, банковская платежная карточка. Документарные операции представлены аккредитивом, банковской гарантией, инкассо и резервным обязательством. Денежные переводы могут быть международные, внутригосударственные и внутрибанковские.

## 2.5 Архитектура программного комплекса

Ключевым принципом создания кросс-платформенных приложений является создание архитектуры, которая предоставляет максимизации для совместного использования кода на разных платформах. Следование приведенным ниже принципам объектно-ориентированного программирования помогает создать хорошо спроектированное приложение:

Инкапсуляция – обеспечение того, что классы и даже уровни архитектуры предоставляют только минимальный *API*, который выполняет необходимые функции, и скрывает детали реализации. На уровне класса это означает, что объекты ведут себя как «Черные квадратики», и для использования кода не нужно знать, как они выполняют свои задачи. На уровне архитектуры это означает реализацию таких шаблонов, как фасадной, которые реализуют упрощенный *API*, координирующий более сложные взаимодействия от имени кода в более абстрактных слоях. Это означает, что код пользовательского интерфейса (например,) должен отвечать только за отображение экранов и принимать входные данные пользователя; и никогда не взаимодействуют с базой данных напрямую. Аналогичным образом, код доступа к данным должен читать и записывать данные только в базу данных, но никогда не взаимодействует напрямую с кнопками или метками.

Полиморфизм — программирование на интерфейс (или абстрактный класс), поддерживающее несколько реализаций, означает, что основной код может быть написан и совместно использоваться на разных платформах, в то же время взаимодействуя с функциями, зависящими от платформы.

Естественным результатом является приложение, смоделированное после реальных или абстрактных сущностей с отдельными логическими слоями. Разделение кода на слои облегчает понимание, тестирование и обслуживание приложений. Ре-

комендуется физически разделить код на каждом слое (в каталогах или даже отдельные проекты для очень больших приложений), а также логически разделить (с использованием пространств имен).

Разрабатываемый программный комплекс содержит *web*-интерфейс для доступа к данным и мобильное приложение – интерфейс для работы с данными, взаимодействует с *web*-интерфейсом для получения и отправки данных. Архитектура представлена на рисунке 2.1.

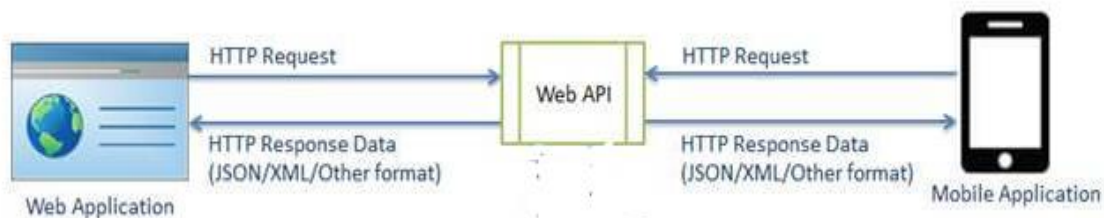


Рисунок 2.1 – Архитектура приложения

В мобильном приложении используется шаблон *MVVM*.

Шаблоны — это установленный способ сбора повторяющихся решений для распространенных проблем.

Модель, представление, *ViewModel* (*MVVM*) — шаблон *Model-View-View-Model* популярен с платформами, поддерживающими привязку данных, например *Xamarin.Forms*. Он был популярным пакетами *SDK* с поддержкой *XAML*, такими как *Windows Presentation Foundation (WPF)* и *Silverlight*, где *ViewModel* выступает в качестве пути взаимодействия между данными (моделью) и пользовательским интерфейсом (представлением) с помощью привязки данных и команд.

### 3 СТРУКТУРА ПРОГРАММНОГО КОМПЛЕКСА

#### 3.1 Структура базы данных.

При исследовании предметной области были выделены следующие сущности:

- «*Users*» – хранит регистрационную информацию пользователей;
- «*Events*» – хранит информацию о мероприятиях;
- «*UserEvents*» – хранит информацию пользователей, которые принимают участие в мероприятии;
- «*UserEventDocuments*» – хранит ссылки на документы об оплате пользователем мероприятия.

Атрибуты сущности «*Users*»:

- *Id* – уникальный номер, однозначно идентифицирующий каждого пользователя;
- *Login* – логин пользователя;
- *Password* – пароль;
- *Firstname* – имя;
- *Surname* – фамилия;
- *Email* – адрес электронной почты;
- *Phone* – телефон;
- *Role* – роль пользователя в системе.

Атрибуты сущности «*Events*»:

- *Id* – уникальный номер, однозначно идентифицирующий каждое мероприятие;
- *Name* – название мероприятия;
- *Description* – описание мероприятия;
- *Total* – общая стоимость мероприятия;
- *UserTotal* – стоимость мероприятия на человека;
- *AuthorId* – идентификатор пользователя создавшего мероприятие.

Атрибуты сущности «*UserEvents*»:

- *Id* – уникальный номер, однозначно идентифицирующий каждого пользователя в мероприятии;
- *UserId* – идентификатор пользователя;
- *EventId* – идентификатор мероприятия;
- *Total* – стоимость мероприятия;
- *Progress* – оплачено пользователем.



Атрибуты сущности «*UserEventDocuments*»:

- *Id* – уникальный номер, однозначно идентифицирующий каждый документ;
- *UserEventId* – идентификатор пользователя в мероприятии;
- *filePath* – путь к документу об оплате.

### 3.2 Структура web-приложения

Структура данного приложения выглядит следующим образом:

- *Dependencies*: все добавленные в проект пакеты и библиотеки;
- *wwwroot*: этот узел (на жестком диске ему соответствует одноименная папка) предназначен для хранения статических файлов – изображений, скриптов *javascript*, файлов *css* и т.д., которые используются приложением. Цель добавления этой папки в проект по сравнению с другими версиями *ASP.NET*, состоит в разграничении доступа к статическим файлам, к которым разрешен доступ со стороны клиента и к которым доступ запрещен;
- *Controllers*: папка для хранения контроллеров, используемых приложением;
- *Models*: каталог для хранения моделей;
- *Views*: каталог для хранения представлений;
- *appsettings.json*: хранит конфигурацию приложения;
- *bower.json*: файл, который управляет клиентскими зависимостями (библиотеки *javascript* и *css*), которые подключаются через менеджер пакетов *Bower*;
- *bundleconfig.json*: файл, который содержит задачи по минификации используемых скриптов и стилей, которые выполняются при построении проекта;
- *Program.cs*: файл, определяющий класс *Program*, который инициализирует и запускает хост с приложением;
- *Startup.cs*: файл, определяющий класс *Startup*, с которого начинается работа приложения. То есть это входная точка в приложение.

Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать. Структура используемых в приложении моделей изображена на рисунке 3.1.

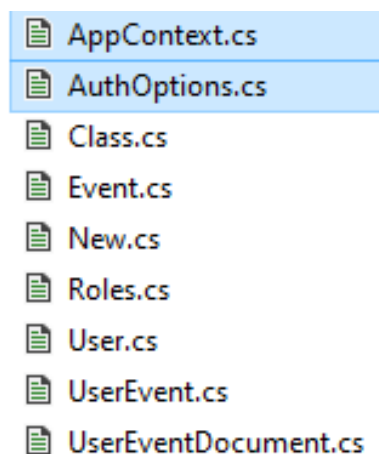


Рисунок 3.1 – Структура моделей

В папке *Models* находятся следующие модели:

- «*AppContext*» – класс контекста данных;
- «*AuthOptions*» – этот класс хранит данные, используемые для аутентификации;
- «*Users*» – этот класс хранит регистрационную информацию пользователей;
- «*Events*» – этот класс хранит информацию о мероприятиях;
- «*UserEvents*» – этот класс хранит информацию пользователях, которые принимают участие в мероприятии;
- «*UserEventDocuments*» – этот класс хранит ссылки на документы об оплате пользователем мероприятия.

Контроллер (англ. *Controller*). Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

На рисунке 3.2 представлена структура всех созданных контроллеров приложения.

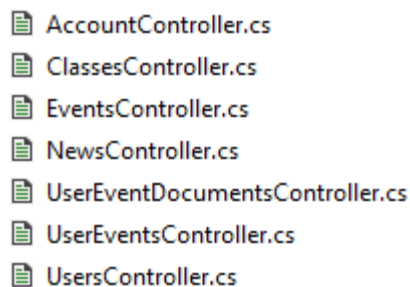


Рисунок 3.2 – Структура контроллеров

В папке *Controllers* находятся контроллеры, которые обрабатывают данные. В контроллерах находятся следующие методы:

*AccountController:*

- *Token()* – *get* метод для проверки токена;
- *GetIdentity()* – *get* метод для авторизации и выдачи токена;

*UsersController:*

- *Get()* – *get* метод для получения списка пользователей;
- *Get(id)* – *get* метод для получения пользователя;
- *Post()* – *post* метод для добавления пользователя;
- *Put()* – *put* метод для обновления пользователя;
- *Delete()* – *deltete* метод для удаления пользователя.

*EventsController:*

- *Get()* – *get* метод для получения списка мероприятий;
- *Get(id)* – *get* метод для получения мероприятия;
- *Post()* – *post* метод для добавления мероприятия;
- *Put()* – *put* метод для обновления мероприятия;
- *Delete()* – *deltete* метод для удаления мероприятия.

*UserEventsController:*

- *Get()* – *get* метод для получения списка пользователей в мероприятии;
- *Get(id)* – *get* метод для получения пользователя в мероприятии;
- *Post()* – *post* метод для добавления пользователя в мероприятии;
- *Put()* – *put* метод для обновления пользователя в мероприятии;
- *Delete()* – *deltete* метод для удаления пользователя в мероприятии.

*UserEventDocumentsController:*

- *Get()* – *get* метод для получения списка документов;
- *Get(id)* – *get* метод для получения документа;
- *Post()* – *post* метод для добавления документа;
- *Put()* – *put* метод для обновления документа;
- *Delete()* – *deltete* метод для удаления документа.

С подробным кодом контроллеров можно ознакомиться в приложении А.

### 3.3 Структура мобильного приложения

Структура данного приложения выглядит следующим образом:

- *Dependencies*: все добавленные в проект пакеты и библиотеки;
- *Models*: каталог для хранения моделей;
- *Views*: каталог для хранения представлений;

- *ViewModels*: каталог для хранения моделей в том виде, в котором они будут отображаться у пользователя;
- *Services*: каталог для хранения классов, которые взаимодействуют с сервером для получения информации;
- *launchSettings.json*: хранит конфигурацию приложения;
- *App.xaml.cs*: файл, определяющий класс, с которого начинается работа приложения. То есть это входная точка в приложение.

Структура используемых в приложении моделей изображена на рисунке 3.2.

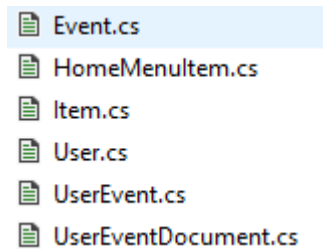


Рисунок 3.2 – Структура моделей

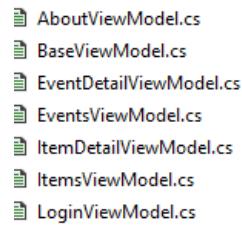
В папке *Models* находятся следующие модели:

- «*Users*» – этот класс хранит регистрационную информацию пользователей;
- «*Events*» – этот класс хранит информацию о мероприятиях;
- «*UserEvents*» – этот класс хранит информацию пользователей, которые принимают участие в мероприятии;
- «*UserEventDocuments*» – этот класс хранит ссылки на документы об оплате пользователем мероприятия.

В папке *ViewModels* находятся следующие классы:

- «*LoginViewModel*» – модель для отображения информации на странице авторизации;
- «*EventsViewModel*» – модель для отображения информации на странице с мероприятиями;
- «*EventDetailViewModel*» – модель для отображения информации на странице с мероприятием.

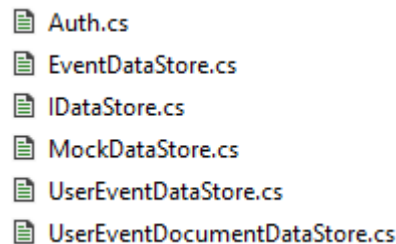
Структура используемых в приложении моделей для отображения изображена на рисунке 3.3.



- AboutViewModel.cs
- BaseViewModel.cs
- EventDetailViewModel.cs
- EventsViewModel.cs
- ItemDetailViewModel.cs
- ItemsViewModel.cs
- LoginViewModel.cs

Рисунок 3.3 – Структура моделей для отображения

В папке «*Services*» хранятся классы, используемые для подключения к *web*-ресурсам, и получения от них необходимой информации. Структура изображена на рисунке 3.4.



- Auth.cs
- EventDataStore.cs
- IDataStore.cs
- MockDataStore.cs
- UserEventDataStore.cs
- UserEventDocumentDataStore.cs

Рисунок 3.4 – Структура сервисов мобильного приложения

### 3.4 Ролевые политики для доступа к ресурсам и функциям системы.

Ролевая модель (*Role-Based Access Control*), разработанная в 1992 году, послужила новым витком эволюции моделей контроля доступа. Ее оформили в виде стандарта *ANSI/INCITS15* в 2004 году. Данная модель является естественным отражением должностных структур организаций и служебных обязанностей пользователей. *RBAC* основана на присвоении пользователям «ролей», представляющих собой множество разрешенных «функций», т.е. пар (действие, объект). Можно сказать, что «роль» аналогична должности в компании [5].

В приложении были разработаны следующие роли:

- родитель: может просматривать доступные для него мероприятия, добавлять документы об оплате;
- член родительского комитета, учитель: могут добавлять мероприятия, просматривать, добавлять документы об оплате, отмечать оплату у пользователей, в мероприятиях, где они являются создателями;
- администратор: может выполнять любые действия, описанные выше, кроме того, может создавать новых пользователей.

### 3.5 Подробный функционал и интерфейс мобильного приложения

При входе в приложение пользователь оказывается на главной странице – странице авторизации, где пользователю предлагается ввести данные для входа в приложение: логин, пароль. Страница авторизации изображена на рисунке 3.5.

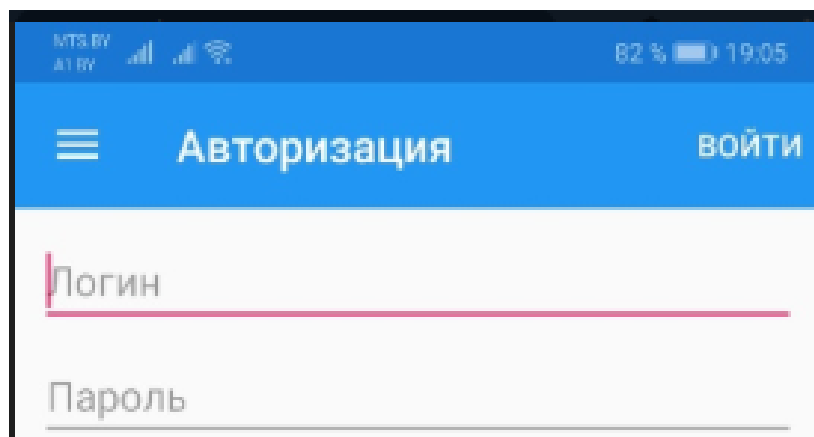


Рисунок 3.5 – Страница авторизации

В приложении разработано два вида меню, первое – для неавторизованного пользователя, второе – для авторизованного пользователя. В первом меню содержатся пункты, отображающие сведения о приложении. Во втором меню кроме пунктов, отображающих сведения о приложении, также содержатся пункты, позволяющие работать с данными («Мероприятия»). Виды меню изображены на рисунках 3.6, 3.7. Для открытия меню необходимо провести по экрану с левой стороны к центру, либо же в левом верхнем углу нажать на иконку «Меню».

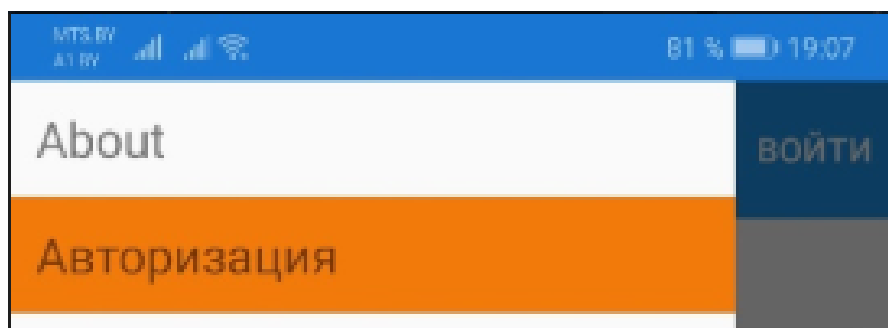


Рисунок 3.6 – Меню неавторизованного пользователя



Рисунок 3.7 – Меню авторизованного пользователя

Если пользователь авторизован, и выбрал пункт меню «Выход», токен аутентификации сотрется и произойдет выход на страницу авторизации. Неавторизованный пользователь не может полностью использовать основной функционал приложения.

При выборе пункта меню «Мероприятия», открывается страница с доступными для пользователя мероприятиями. Если выбрать одно из них, и нажать на него на экран будет выведена информация о нем, название, описание, сумму, которую необходимо заплатить и уже оплаченная сумма. Страница с мероприятиями изображена на рисунке 3.8.

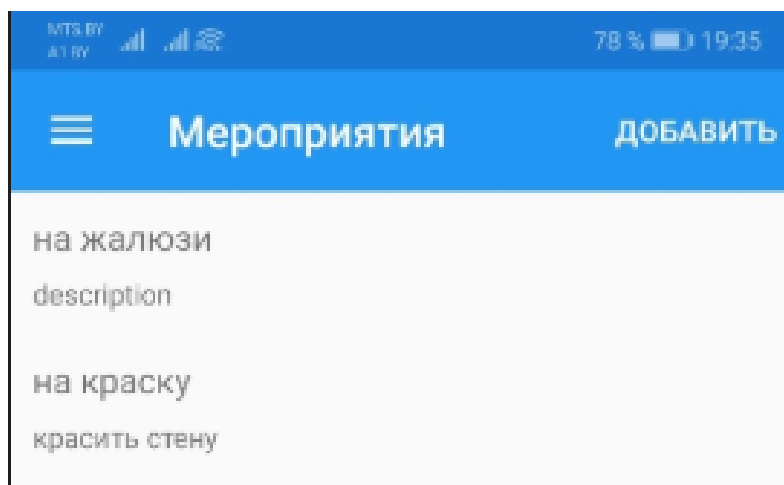


Рисунок 3.8 – Страница с мероприятиями

Страница о подробной информации мероприятия изображена на рисунке 3.9.

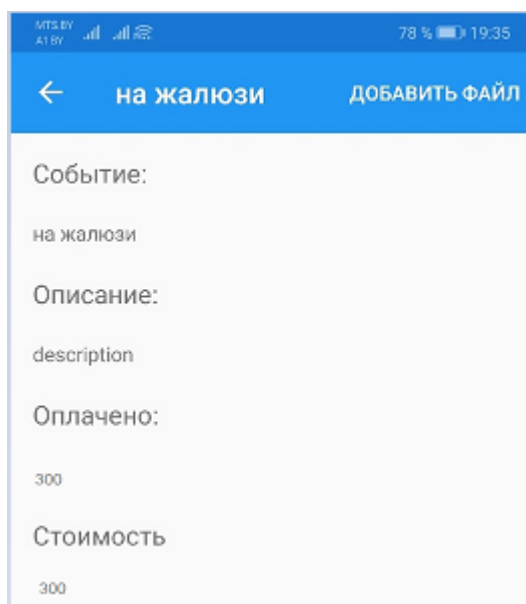


Рисунок 3.9 – Подробная информация о мероприятии

При нажатии на кнопку «Добавить файл» происходит перенаправление на файлы устройства, пользователю необходимо выбрать файл об оплате мероприятия, после чего администратор задания должен проверить файл и изменить значения об оплате пользователя. На рисунке 3.10 изображено действие устройства. Которое происходит при нажатии на кнопку «Добавить файл»

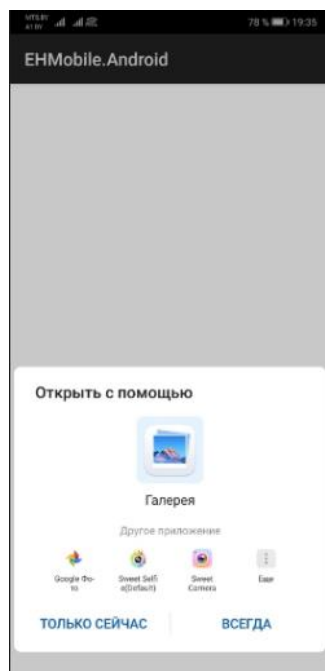


Рисунок 3.10 – Добавить файл



## ЗАКЛЮЧЕНИЕ

В процессе прохождения технологической практики ознакомилась с предприятием ОДО «Альфа», его организационной структурой, подразделениями, информационной системой, видами деятельности, используемыми технологиями, принципами управления и работы.

Кроме того, были закреплены, расширены, углублены и систематизированы теоретические знания в области создания *мобильных* приложений. Также были приобретены навыки проектирования и конструирования информационных систем.

Согласно теме, выданной руководителем практики со стороны предприятия, были изучены *C#* библиотека *Xamarin.Forms* служащая для разработки мобильных приложений. Также была изучена технология *ASP.NET WEB API APP*, служащая для разработки *web*-приложений. По индивидуальному заданию, выданному руководителем практики со стороны предприятия, было разработано приложение с использованием *C#* библиотек.

### Список использованных источников

1. Альфа-Гомель [Электронный ресурс]. – Режим доступа: <http://www.alfagomel.by/> – Дата доступа: 17.04.2020.
2. Указ Президента Республики Беларусь от 16 октября 2014 г. № 493 «О развитии безналичных расчетов».
3. Инструкция об организации исполнения платежей с текущих (расчетных) счетов в белорусских рублях в очередности, установленной законодательством, утвержденная постановлением Правления Нацбанка РБ от 29.03.2001 № 63 (с изменениями и дополнениями) («НРПА РБ», 2001, № 43).
4. Закон Республики Беларусь “Об электронном документе” от 28.12.2009 г. №113-3 [Электронный ресурс] / Законодательство Республики Беларусь. – Минск. – Режим доступа: [pravoby.info](http://pravoby.info). – Дата доступа: 17.04.2020.
5. Role-Based Access Controls / David F. Ferraiolo D. Richard Kuhn. – [б.м.]: National Institute of Standards and Technology, 1992 г.

**ПРИЛОЖЕНИЕ А**  
(обязательное)  
**Листинг программы**

```
namespace EducationHelper.Controllers
{
    public class AccountController : ControllerBase
    {
        Models.AppContext db;
        public AccountController(Models.AppContext context)
        {
            db = context;
        }
        [HttpPost("/token")]
        public IActionResult Token(string username, string password)
        {
            var identity = GetIdentity(username, password);
            if (identity == null)
            {
                return BadRequest(new { errorText = "Invalid username or password." });
            }

            var now = DateTime.UtcNow;
            // создаем JWT-токен
            IdentityModelEventSource.ShowPII = true;
            var jwt = new JwtSecurityToken(
                issuer: AuthOptions.ISSUER,
                audience: AuthOptions.AUDIENCE,
                notBefore: now,
                claims: identity.Claims,
                expires: now.Add(TimeSpan.FromMinutes(AuthOptions.LIFETIME)),
                signingCredentials: new SigningCredentials(AuthOptions.GetSymmetricSecurityKey(), SecurityAlgorithms.HmacSha256));
            var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);
            var response = new
            {
                access_token = encodedJwt,
                username = identity.Name
            };
            return new ObjectResult(response);
        }

        [Authorize]
        [Route("/checkAuth")]
    }
}
```

```

public IActionResult Token()
{
    var r = db.Users.FirstOrDefault(p => p.Login == User.Identity.Name).Role;
    return Ok(r);
}
private ClaimsIdentity GetIdentity(string username, string password)
{
    User user = db.Users.FirstOrDefault(x => x.Login == username && x.Password == password);
    if (user != null)
    {
        var claims = new List<Claim>
        {
            new Claim(ClaimsIdentity.DefaultNameClaimType, user.Login),
            new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role)
        };
        ClaimsIdentity claimsIdentity =
            new ClaimsIdentity(claims, "Token", ClaimsIdentity.DefaultNameClaimType,
                ClaimsIdentity.DefaultRoleClaimType);
        return claimsIdentity;
    }

    // если пользователя не найдено
    return null;
} }}
namespace EducationHelper.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class EventsController : ControllerBase
    {
        Models.AppContext db;
        public EventsController(Models.AppContext context)
        {
            db = context;
        }

        [HttpGet]
        [Authorize]
        public ActionResult<IEnumerable<Event>> Get()
        {
            int role = Convert.ToInt32(db.Users.FirstOrDefault(p => p.Login == User.Identity.Name).Role);

```

```

List<Event> result = new List<Event>();

switch (role)
{
    case 1 : result = db.Events.ToList(); break;
    case 2 :
    {
        int userid = db.Users.FirstOrDefault(p => p.Login == User.Identity.Name).Id;
        List<UserEvent> userEvents = db.UserEvents.Where(p => p.UserId ==
userid).ToList();
        foreach (UserEvent ue in userEvents)
        {
            result.Add(db.Events.FirstOrDefault(p => p.Id == ue.EventId));
        }
    } break;
    case 3:
    {
        int userid = db.Users.FirstOrDefault(p => p.Login == User.Identity.Name).Id;
        List<UserEvent> userEvents = db.UserEvents.Where(p => p.UserId ==
userid).ToList();
        foreach (UserEvent ue in userEvents)
        {
            result.Add(db.Events.FirstOrDefault(p => p.Id == ue.EventId));
        }
        foreach (Event e in db.Events.Where(p => p.AuthorId == userid))
        {
            result.Add(e);
        }
    } break;
}

return result;
}
// GET /5
[HttpGet("{id}")]
public async Task<ActionResult<Event>> Get(int id)
{
    Event res = await db.Events.FirstOrDefaultAsync(x => x.Id == id);
    if (res == null)
        return NotFound();
    return new ObjectResult(res);
}
// POST

```

```

[HttpPost]
public async Task<ActionResult<Event>> Post(Event _event)
{
    if (_event == null)
    {
        return BadRequest();
    }

    db.Events.Add(_event);
    await db.SaveChangesAsync();
    return Ok(_event);
}
// PUT
[HttpPut]
public async Task<ActionResult<Event>> Put(Event _event)
{
    if (_event == null)
    {
        return BadRequest();
    }
    if (!db.Events.Any(x => x.Id == _event.Id))
    {
        return NotFound();
    }
    db.Update(_event);
    await db.SaveChangesAsync();
    return Ok(_event);
}
// DELETE
[HttpDelete("{id}")]
public async Task<ActionResult<Event>> Delete(int id)
{
    Event _event = db.Events.FirstOrDefault(x => x.Id == id);
    if (_event == null)
    {
        return NotFound();
    }
    db.Events.Remove(_event);
    await db.SaveChangesAsync();
    return Ok(_event);
}
}
}}
namespace EducationHelper.Controllers
{
    [Route("[controller]")]

```

```

[ApiController]
public class UserEventsController : ControllerBase
{
    Models.AppContext db;
    public UserEventsController(Models.AppContext context)
    {
        db = context;
    }

    [HttpGet]
    public ActionResult<IEnumerable<UserEvent>> Get()
    {
        return db.UserEvents.ToList();
    }

    // GET /5
    [HttpGet("{id}")]
    [Authorize]
    public async Task<ActionResult<UserEvent>> Get(int id)
    {
        UserEvent res = await db.UserEvents.FirstOrDefaultAsync(x => x.EventId == id &&
x.UserId == Convert.ToInt32(db.Users.FirstOrDefault(p => p.Login == User.Iden-
tity.Name).Id));
        if (res == null)
            return NotFound();
        return new ObjectResult(res);
    }

    // POST
    [HttpPost]
    public async Task<ActionResult<UserEvent>> Post(UserEvent userEvent)
    {
        if (userEvent == null)
        {
            return BadRequest();
        }

        db.UserEvents.Add(userEvent);
        await db.SaveChangesAsync();
        return Ok(userEvent);
    }

    // PUT
    [HttpPut]
    public async Task<ActionResult<UserEvent>> Put(UserEvent userEvent)

```

```

    {
        if (userEvent == null)
        {
            return BadRequest();
        }
        if (!db.UserEvents.Any(x => x.Id == userEvent.Id))
        {
            return NotFound();
        }

        db.Update(userEvent);
        await db.SaveChangesAsync();
        return Ok(userEvent);
    }
    // DELETE
    [HttpDelete("{id}")]
    public async Task<ActionResult<UserEvent>> Delete(int id)
    {
        UserEvent userEvent = db.UserEvents.FirstOrDefault(x => x.Id == id);
        if (userEvent == null)
        {
            return NotFound();
        }
        db.UserEvents.Remove(userEvent);
        await db.SaveChangesAsync();
        return Ok(userEvent);
    }
}

namespace EducationHelper.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class UsersController : ControllerBase
    {
        Models.AppContext db;
        public UsersController(Models.AppContext context)
        {
            db = context;
        }

        [HttpGet]
        public ActionResult<IEnumerable<User>> Get()
        {
            return db.Users.ToList();
        }
    }
}

```



```

    }

// GET /5
[HttpGet("{id}")]
public async Task<ActionResult<User>> Get(int id)
{
    User res = await db.Users.FirstOrDefaultAsync(x => x.Id == id);
    if (res == null)
        return NotFound();
    return new ObjectResult(res);
}

// POST
[HttpPost]
public async Task<ActionResult<User>> Post(User _user)
{
    if (_user == null)
    {
        return BadRequest();
    }

    db.Users.Add(_user);
    await db.SaveChangesAsync();
    return Ok(_user);
}

// PUT
[HttpPut]
public async Task<ActionResult<User>> Put(User _user)
{
    if (_user == null)
    {
        return BadRequest();
    }
    if (!db.Users.Any(x => x.Id == _user.Id))
    {
        return NotFound();
    }

    db.Update(_user);
    await db.SaveChangesAsync();
    return Ok(_user);
}

```

```

// DELETE
[HttpDelete("{id}")]
public async Task<ActionResult<User>> Delete(int id)
{
    User _user = db.Users.FirstOrDefault(x => x.Id == id);
    if (_user == null)
    {
        return NotFound();
    }
    db.Users.Remove(_user);
    await db.SaveChangesAsync();
    return Ok(_user);
} }}

namespace EducationHelper.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class UserEventDocumentsController : ControllerBase
    {
        Models.AppContext db;
        public UserEventDocumentsController(Models.AppContext context)
        {
            db = context;
        }
        [HttpGet]
        public ActionResult<IEnumerable<UserEventDocument>> Get()
        {
            return db.UserEventDocuments.ToList();
        }
        // GET /5
        [HttpGet("{id}")]
        public async Task<ActionResult<UserEventDocument>> Get(int id)
        {
            UserEventDocument res = await db.UserEventDocuments.FirstOrDefaultAsync(x =>
x.Id == id);
            if (res == null)
                return NotFound();
            return new ObjectResult(res);
        }
        // POST
        [HttpPost]
        public async Task<ActionResult<UserEventDocument>> Post(UserEventDocument us-
erEventDocument)

```

```

    {
        if (userEventDocument == null)
        {
            return BadRequest();
        }
        db.UserEventDocuments.Add(userEventDocument);
        await db.SaveChangesAsync();
        return Ok(userEventDocument);
    }
    // PUT
    [HttpPut]
    public async Task<ActionResult<UserEventDocument>> Put(UserEventDocument userEventDocument)
    {
        if (userEventDocument == null)
        {
            return BadRequest();
        }
        if (!db.UserEventDocuments.Any(x => x.Id == userEventDocument.Id))
        {
            return NotFound();
        }
        db.Update(userEventDocument);
        await db.SaveChangesAsync();
        return Ok(userEventDocument);
    }
    // DELETE
    [HttpDelete("{id}")]
    public async Task<ActionResult<UserEventDocument>> Delete(int id)
    {
        UserEventDocument userEventDocument = db.UserEventDocuments.FirstOrDefault(x
=> x.Id == id);
        if (userEventDocument == null)
        {
            return NotFound();
        }
        db.UserEventDocuments.Remove(userEventDocument);
        await db.SaveChangesAsync();
        return Ok(userEventDocument);
    } }
namespace EducationHelper
{
    public class Startup
    {

```

```

public Startup(IConfiguration configuration)
{
    Configuration = configuration;
}

public IConfiguration Configuration { get; }

string connection = Configuration.GetConnectionString("DbConnection");
services.AddDbContext<Models.AppContext>(options => options.UseMySQL(connection));

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.RequireHttpsMetadata = false;
        options.TokenValidationParameters = new TokenValidationParameters
        {
            // укзывает, будет ли валидироваться издатель при валидации токена
            ValidateIssuer = true,
            // строка, представляющая издателя
            ValidIssuer = AuthOptions.ISSUER,

            // будет ли валидироваться потребитель токена
            ValidateAudience = true,
            // установка потребителя токена
            ValidAudience = AuthOptions.AUDIENCE,
            // будет ли валидироваться время существования
            ValidateLifetime = true,

            // установка ключа безопасности
            IssuerSigningKey = AuthOptions.GetSymmetricSecurityKey(),
            // валидация ключа безопасности
            ValidateIssuerSigningKey = true,
        };
    });

services.AddControllers();
services.AddControllersWithViews();
}

// This method gets called by the runtime. Use this method to configure the HTTP request
// pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseDeveloperExceptionPage();
    app.UseDefaultFiles();
    app.UseStaticFiles();
    app.UseRouting();
}

```

```

        // app.UseHttpsRedirection();
        app.UseAuthentication();
        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}

namespace EducationHelper.Models
{
    public class AppContext : DbContext
    {
        public DbSet<User> Users { get; set; }
        public DbSet<Class> Classes { get; set; }
        public DbSet<New> News { get; set; }
        public DbSet<Event> Events { get; set; }
        public DbSet<UserEvent> UserEvents { get; set; }
        public DbSet<UserEventDocument> UserEventDocuments { get; set; }

        public AppContext(DbContextOptions<AppContext> options) : base(options) { }
    }
}

namespace EducationHelper.Models
{
    public class AuthOptions
    {
        public const string ISSUER = "MyAuthServer"; // издатель токена
        public const string AUDIENCE = "MyAuthClient"; // потребитель токена
        public const string KEY = "shsshsshsshsshsshsshsshsshsshsshsshsshsshsshsshsshs"; //
        ключ для шифрации
        public const int LIFETIME = 120; // время жизни токена - 120 минута
        public static SymmetricSecurityKey GetSymmetricSecurityKey()
        {
            return new SymmetricSecurityKey(Encoding.ASCII.GetBytes(KEY));
        }
    }
}

namespace Common.Models
{
    public class Event
    {
        public int Id { get; set; }
        public int Total { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public int AuthorId { get; set; }
        public int UserTotal { get; set; }
    }
}

```

```

    }}
namespace Common.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }
        public string Firstname { get; set; }
        public string Surname { get; set; }
        public string Email { get; set; }
        public string Phone { get; set; }
        public string Role { get; set; }
    }}
namespace Common.Models
{
    public class UserEvent
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public int EventId { get; set; }
        public int Progress { get; set; }
        public int Total { get; set; }
    }}
namespace Common.Models
{
    public class UserEventDocument
    {
        public int Id { get; set; }
        public int UserEventId { get; set; }
        public string FilePath { get; set; }
    }}
namespace EHMobile.Services
{
    public static class Auth
    {
        static int role;
        public static int Role {
            get { return role; }
            set {
                if (role != value)
                {
                    role = value;
                    RoleChanged();
                }
            }
        }
    }
}

```

```

        }
    }
}

public delegate void RoleChangedDelegate();
public static event RoleChangedDelegate RoleChanged;
public static async Task<int> GetRole()
{
    WebRequest request = WebRequest.Create("http://172.20.10.2:8888/checkAuth");
    //request2.Headers.Set("Accept", "application/json");
    ((HttpWebRequest)request).Accept = "application/json";
    request.Headers.Set("Authorization", "Bearer " + (string)App.Current.Properties["Token"]);
    WebResponse response = await request.GetResponseAsync();
    string role;
    role = "";
    using (Stream stream = response.GetResponseStream())
    {
        using (StreamReader reader = new StreamReader(stream))
        {
            role += reader.ReadToEnd();
        }
        response.Close();
    }
    Role = Convert.ToInt32(role.Replace("\"", ""));

    return Role;
}

public static async Task<int> Login(string username, string password)
{
    WebRequest request = WebRequest.Create("http://172.20.10.2:8888/token");
    request.Method = "POST";
    string data = "username=" + username + "&password=" + password;
    byte[] byteArray = System.Text.Encoding.UTF8.GetBytes(data);
    request.ContentType = "application/x-www-form-urlencoded";
    request.ContentLength = byteArray.Length;
    using (Stream dataStream = request.GetRequestStream())
    {
        dataStream.Write(byteArray, 0, byteArray.Length);
    }
    try
    {
        WebResponse response = await request.GetResponseAsync();
        string token = "";
        using (Stream stream = response.GetResponseStream())
        {
            using (StreamReader reader = new StreamReader(stream))
            {
                token += reader.ReadToEnd();
            }
        }
        //App.Current.Properties.Add("Token", "");
    }
}

```

```

        App.Current.Properties["Token"] = token.Split(':')[1].Split(',')[0].Replace("\"", "");
        response.Close();

        return await GetRole();
    }
    catch
    {
        return -1;
    }
}

namespace EHMobile.Services
{
    public class EventDataStore : IDataStore<Event>
    {
        readonly List<Event> items;
        const string Url = "http://172.20.10.2:8888/events";

        public EventDataStore()
        {
            items = new List<Event>()
            {
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" }
            };
        }

        // настройка клиента
        private HttpClient GetClient()
        {
            HttpClient client = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return client;
        }

        public async Task<bool> AddItemAsync(Event item)
        {
            items.Add(item);
        }
    }
}

```



```

        return await Task.FromResult(true);
    }

    public async Task<bool> UpdateItemAsync(Event item)
    {
        var oldItem = items.Where((Event arg) => arg.Id == item.Id).FirstOrDefault();
        items.Remove(oldItem);
        items.Add(item);

        return await Task.FromResult(true);
    }

    public async Task<bool> DeleteItemAsync(int id)
    {
        var oldItem = items.Where((Event arg) => arg.Id == id).FirstOrDefault();
        items.Remove(oldItem);

        return await Task.FromResult(true);
    }

    public async Task<Event> GetItemAsync(int id)
    {
        return await Task.FromResult(items.FirstOrDefault(s => s.Id == id));
    }

    public async Task<IEnumerable<Event>> GetItemsAsync(bool forceRefresh = false)
    {
        ((HttpRequest)request).Accept = "application/json";
        request.Headers.Set("Authorization", "Bearer " + (string)App.Current.Properties["Token"]);
        WebResponse response = await request.GetResponseAsync();

        string result = "";
        using (Stream stream = response.GetResponseStream())
        {
            using (StreamReader reader = new StreamReader(stream))
            {
                result += reader.ReadToEnd();
            }
        }
        response.Close();

        //HttpClient client = GetClient();
        //string result = await client.GetStringAsync(Url);
        return JsonConvert.DeserializeObject<IEnumerable<Event>>(result);
    }

```

```

        //return await Task.FromResult(items);
    } }
namespace EHMobile.Services
{
    public interface IDataStore<T>
    {
        Task<bool> AddItemAsync(T item);
        Task<bool> UpdateItemAsync(T item);
        Task<bool> DeleteItemAsync(int id);
        Task<T> GetItemAsync(int id);
        Task<IEnumerable<T>> GetItemsAsync(bool forceRefresh = false);
    }
}
namespace EHMobile.Services
{
    public class MockDataStore : IDataStore<Item>
    {
        readonly List<Item> items;

        public MockDataStore()
        {
            items = new List<Item>()
            {
                new Item { Id = Guid.NewGuid().ToString(), Text = "First item", Description="This is
an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Second item", Description="This
is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Third item", Description="This
is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Fourth item", Description="This
is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Fifth item", Description="This
is an item description." },
                new Item { Id = Guid.NewGuid().ToString(), Text = "Sixth item", Description="This
is an item description." }
            };
        }

        public async Task<bool> AddItemAsync(Item item)
        {
            items.Add(item);

            return await Task.FromResult(true);
        }

        public async Task<bool> UpdateItemAsync(Item item)
        {

```

```

        var oldItem = items.Where((Item arg) => arg.Id == item.Id).FirstOrDefault();
        items.Remove(oldItem);
        items.Add(item);
        return await Task.FromResult(true);
    }
    public async Task<bool> DeleteItemAsync(int id)
    {
        var oldItem = items.Where((Item arg) => arg.Id == id.ToString()).FirstOrDefault();
        items.Remove(oldItem);
        return await Task.FromResult(true);
    }
    public async Task<Item> GetItemAsync(int id)
    {
        return await Task.FromResult(items.FirstOrDefault(s => s.Id == id.ToString()));
    }
    public async Task<IEnumerable<Item>> GetItemsAsync(bool forceRefresh = false)
    {
        return await Task.FromResult(items);
    }
}
namespace EHMobile.Services
{
    public class UserEventDataStore : IDataStore<UserEvent>
    {
        readonly List<UserEvent> items;
        const string Url = "http://172.20.10.2:8888/userevents";
        public UserEventDataStore()
        {
            items = new List<UserEvent>();
        }
        // настройка клиента
        private HttpClient GetClient()
        {
            HttpClient client = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return client;
        }
        public async Task<bool> AddItemAsync(UserEvent item)
        {
            items.Add(item);
            return await Task.FromResult(true);
        }
        public async Task<bool> UpdateItemAsync(UserEvent item)
        {
            var oldItem = items.Where((UserEvent arg) => arg.Id == item.Id).FirstOrDefault();

```

```

        items.Remove(oldItem);
        items.Add(item);
        return await Task.FromResult(true);
    }
    public async Task<bool> DeleteItemAsync(int id)
    {
        var oldItem = items.Where((UserEvent arg) => arg.Id == id).FirstOrDefault();
        items.Remove(oldItem);
        return await Task.FromResult(true);
    }
    public async Task<UserEvent> GetItemAsync(int id)
    {
        WebRequest request = WebRequest.Create("http://172.20.10.2:8888/userevents/get/" +
id);
        ((HttpWebRequest)request).Accept = "application/json";
        request.Headers.Set("Authorization", "Bearer " + (string)App.Current.Properties["To-
ken"]);
        WebResponse response = await request.GetResponseAsync();

        string result = "";
        using (Stream stream = response.GetResponseStream())
        {
            using (StreamReader reader = new StreamReader(stream))
            {
                result += reader.ReadToEnd();
            }
        }
        response.Close();
        return JsonConvert.DeserializeObject<UserEvent>(result);
    }

    public async Task<IEnumerable<UserEvent>> GetItemsAsync(bool forceRefresh = false)
    {
        ((HttpWebRequest)request).Accept = "application/json";
        request.Headers.Set("Authorization", "Bearer " + (string)App.Current.Properties["To-
ken"]);
        WebResponse response = await request.GetResponseAsync();

        string result = "";
        using (Stream stream = response.GetResponseStream())
        {
            using (StreamReader reader = new StreamReader(stream))
            {
                result += reader.ReadToEnd();
            }
        }
    }

```

```

        }
    }
    response.Close();

    //HttpClient client = GetClient();
    //string result = await client.GetStringAsync(Url);
    return JsonConvert.DeserializeObject<IEnumerable<UserEvent>>(result);
    //return await Task.FromResult(items);
}
}}
namespace EHMobile.Services
{
    public class UserEventDocumentDataStore : IDataStore<Event>
    {
        readonly List<Event> items;

        public UserEventDocumentDataStore()
        {
            items = new List<Event>()
            {
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" },
                new Event { Name = "На жалюзи", Total=500, AuthorId = 1, Description="hhhh" }
            };
        }

        public async Task<bool> AddItemAsync(Event item)
        {
            items.Add(item);

            return await Task.FromResult(true);
        }

        public async Task<bool> UpdateItemAsync(Event item)
        {
            var oldItem = items.Where((Event arg) => arg.Id == item.Id).FirstOrDefault();
            items.Remove(oldItem);
            items.Add(item);

            return await Task.FromResult(true);
        }

        public async Task<bool> DeleteItemAsync(int id)
        {
            var oldItem = items.Where((Event arg) => arg.Id == id).FirstOrDefault();

```

```

        items.Remove(oldItem);

        return await Task.FromResult(true);
    }
    public async Task<Event> GetItemAsync(int id)
    {
        return await Task.FromResult(items.FirstOrDefault(s => s.Id == id));
    }
    public async Task<IEnumerable<Event>> GetItemsAsync(bool forceRefresh = false)
    {
        return await Task.FromResult(items);
    }
}
namespace EHMobile.ViewModels
{
    public class BaseViewModel : INotifyPropertyChanged
    {
        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
}}
namespace EHMobile.ViewModels
{
    public class EventDetailViewModel : BaseViewModel
    {
        public IDataStore<UserEvent> DataStore => DependencyService.Get<IDataStore<UserEvent>>();
        public Event Item { get; set; }
        public UserEvent UserEvent { get; set; }
        public List<UserEventDocument> UserEventDocuments { get; set; }
        public Command LoadItemsCommand { get; set; }
        public EventDetailViewModel(Event item = null, UserEvent userEvent = null, List<UserEventDocument> userEventDocuments = null)
        {
            Title = item?.Name;
            Item = item;
            UserEvent = userEvent;
            UserEventDocuments = userEventDocuments != null ? userEventDocuments : new List<UserEventDocument>();
            LoadItemsCommand = new Command(async () => await ExecuteLoadItemsCommand());
        }
        async Task ExecuteLoadItemsCommand()
        {
            IsBusy = true;

            try
            {
                UserEvent = await DataStore.GetItemAsync(Item.Id);
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex);
            }
            finally
            {
                IsBusy = false;
            }
        }
    }
}
namespace EHMobile.ViewModels
{

```

```

public class EventsViewModel : BaseViewModel
{
    public IDataStore<Event> DataStore => DependencyService.Get<IDataStore<Event>>();
    public ObservableCollection<Event> Items { get; set; }
    public Command LoadItemsCommand { get; set; }
    public EventsViewModel()
    {
        Title = "Мероприятия";
        Items = new ObservableCollection<Event>();
        LoadItemsCommand = new Command(async () => await ExecuteLoadItemsCom-
mand());
        /*MessagingCenter.Subscribe<NewEventPage, Event>(this, "AddItem", async (obj,
item) =>
        {
            var newItem = item as Event;
            Items.Add(newItem);
            await DataStore.AddItemAsync(newItem);
        });*/
    }
    async Task ExecuteLoadItemsCommand()
    {
        IsBusy = true;
        try
        {
            Items.Clear();
            var items = await DataStore.GetItemsAsync(true);
            foreach (var item in items)
            {
                Items.Add(item);
            }
        }
        catch (Exception ex)
        {
            Debug.WriteLine(ex);
        }
        finally
        {
            IsBusy = false;
        }
    }
}

namespace EHMobile.ViewModels
{
    public class LoginViewModel : BaseViewModel
    {
        public string Login { get; set; }
        public string Password { get; set; }
    }
}

```



```

        public LoginViewModel()
        {
            Title = "Авторизация";
        }
    }
}

namespace EHMobile.Views
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class EventDetailPage : ContentPage
    {
        EventDetailViewModel viewModel;

        public EventDetailPage(EventDetailViewModel viewModel)
        {
            InitializeComponent();

            BindingContext = this.viewModel = viewModel;
        }

        public EventDetailPage()
        {
            InitializeComponent();

            var item = new Event
            {
                Name = "Item 1",
                Description = "This is an item description."
            };
            var userEvent = new UserEvent
            {
                Total = 0,
                Progress = 0
            };
            viewModel = new EventDetailViewModel(item, userEvent);
            BindingContext = viewModel;
        }

        async void AddFile_Clicked(object sender, EventArgs args)
        {
            Image img = new Image();
            if (CrossMedia.Current.IsPickPhotoSupported)
            {
                MediaFile photo = await CrossMedia.Current.PickPhotoAsync();
                img.Source = ImageSource.FromFile(photo.Path);
            }
        }
    }
}

```

```

        var uri = new Uri(string.Format("E://Files/Upload/", string.Empty));
        var content = new MultipartFormDataContent();

        content.Add(new StreamContent(photo.GetStream()),
            "\"file\"",
            $"{photo.Path}\"");

        var httpClient = new HttpClient();
        var httpResponseMessage = await httpClient.PostAsync(uri, content);
    }    }    }}

namespace EHMobile.Views
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class EventsPage : ContentPage
    {
        EventsViewModel viewModel;
        public EventsPage()
        {
            InitializeComponent();
            BindingContext = viewModel = new EventsViewModel();
        }
        async void OnItemSelected(object sender, EventArgs args)
        {
            var layout = (BindableObject)sender;
            var item = (Event)layout.BindingContext;
            await Navigation.PushAsync(new EventDetailPage(new EventDetailView-
Model(item)));
        }
        async void AddItem_Clicked(object sender, EventArgs e)
        {
            if(await Auth.GetRole() == 1 || await Auth.GetRole() == 3)
                await Navigation.PushModalAsync(new NavigationPage(new NewEventPage()));
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();

            if (viewModel.Items.Count == 0)
                viewModel.IsBusy = true;
        }    }}

```

```

namespace EHMobile.Views
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class LoginPage : ContentPage
    {
        LoginViewModel viewModel;

        public LoginPage()
        {
            InitializeComponent();

            BindingContext = viewModel = new LoginViewModel();
        }

        async void Login(object sender, EventArgs e)
        {
            if (await Auth.Login(viewModel.Login, viewModel.Password) != -1)
            {
                await Navigation.PushAsync(new AboutPage());
            }
        }

        void e_loginChanged(object sender, EventArgs e)
        {
            viewModel.Login = ((Editor)sender).Text;
        }

        void e_passChanged(object sender, EventArgs e)
        {
            viewModel.Password = ((Editor)sender).Text;
        }
    }
}

namespace EHMobile.Views
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class MainPage : MasterDetailPage
    {
        Dictionary<int, NavigationPage> MenuPages = new Dictionary<int, NavigationPage>();

        public MainPage()
        {
            InitializeComponent();
            MasterBehavior = MasterBehavior.Popover;
        }
    }
}

```

```

public async Task NavigateFromMenu(int id)
{
    if (!MenuPages.ContainsKey(id))
    {
        switch (id)
        {
            case (int)MenuItemType.Login:
                MenuPages.Add(id, new NavigationPage(new LoginPage()));
                break;
            //case (int)MenuItemType.Browse:
            //    MenuPages.Add(id, new NavigationPage(new ItemsPage()));
            //    break;
            case (int)MenuItemType.Events:
                MenuPages.Add(id, new NavigationPage(new EventsPage()));
                break;
            case (int)MenuItemType.About:
                MenuPages.Add(id, new NavigationPage(new AboutPage()));
                break;
            case (int)MenuItemType.Logout:
                {
                    App.Current.Properties["Token"] = "";
                    Auth.Role = -1;
                    MenuPages.Add(id, new NavigationPage(new LoginPage()));
                }
                break;
        }
    }
    var newPage = MenuPages[id];

    if (newPage != null && Detail != newPage)
    {
        Detail = newPage;
        if (Device.RuntimePlatform == Device.Android)
            await Task.Delay(100);

        IsPresented = false;
    }
}

namespace EHMobile.Views
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class MenuPage : ContentPage
    {
        MainPage RootPage { get => Application.Current.MainPage as MainPage; }
        List<HomeMenuItem> menuItems;
    }
}

```

```

public MenuPage()
{
    InitializeComponent();

    menuItems = new List<HomeMenuItem>();
    ChangeMenu();
    Auth.RoleChanged += ChangeMenu;
    //ListViewMenu.SelectedItem = menuItems[0];
    ListViewMenu.ItemSelected += async (sender, e) =>
    {
        if (e.SelectedItem == null)
            return;
        var id = (int)((HomeMenuItem)e.SelectedItem).Id;
        await RootPage.NavigateFromMenu(id);
    };
}
async void ChangeMenu()
{
    menuItems.Clear();
    //menuItems.Add(new HomeMenuItem { Id = MenuItemType.Browse, Title = "Browse"
});
    menuItems.Add(new HomeMenuItem { Id = MenuItemType.About, Title = "About" });
    if (Auth.Role == -1 || Auth.Role == 0)
    {
        menuItems.Add(new HomeMenuItem { Id = MenuItemType.Login, Title =
"Авторизация" });
    }
    else
    {
        menuItems.Add(new HomeMenuItem { Id = MenuItemType.Events, Title =
"Мероприятия" });
        menuItems.Add(new HomeMenuItem { Id = MenuItemType.Logout, Title = "Выход"
});
    }
    ListViewMenu.ItemsSource = menuItems;
}
}
namespace EHMmobile.Views
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class NewEventPage : ContentPage
    {
        public Event Item { get; set; }
        public NewEventPage()

```

```

{
    InitializeComponent();

    Item = new Event
    {
        Name = "Event",
        Description = "This is an item description.",
        Total = 20000,
        UserTotal = 0
    };
    BindingContext = this;
}
async void Save_Clicked(object sender, EventArgs e)
{
    MessagingCenter.Send(this, "AddItem", Item);
    await Navigation.PopModalAsync();
}
async void Cancel_Clicked(object sender, EventArgs e)
{
    await Navigation.PopModalAsync();
}  }}

```