Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

|  |  |  |  |
| --- | --- | --- | --- |
| Институт: | ИВТИ | Кафедра: | Вычислительных машин, систем и сетей |
| Направление подготовки: | | 09.03.01 Информатика и вычислительная техника | |

**ОТЧЕТ по практике**

|  |  |
| --- | --- |
| **Наименование практики:** | *Производственная практика: преддипломная практика* |

**СТУДЕНТ**

|  |  |
| --- | --- |
|  | / Саутов Д. С. / |
| *(подпись)* | (*Фамилия и инициалы*) |

|  |  |
| --- | --- |
| Группа | А-12-18 |
|  | *(номер учебной группы)* |

**ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ПРАКТИКЕ**

|  |
| --- |
|  |
| *(зачтено, не зачтено)* |

|  |  |
| --- | --- |
|  | / / |
| *(подпись)* | (*Фамилия и инициалы члена комиссии*) |

|  |  |
| --- | --- |
|  | / / |
| *(подпись)* | (*Фамилия и инициалы члена комиссии*) |

Оглавление

[ВВЕДЕНИЕ 3](#_Toc103866423)

[1. Задание 4](#_Toc103866424)

[1.1. Описание предметной области 4](#_Toc103866425)

[Система GPS (или подобные ей ГЛОНАСС, GALILEO) 5](#_Toc103866426)

[Геолокация на основе вышек операторов сотовой связи 6](#_Toc103866427)

[1.2. Постановка задачи 7](#_Toc103866428)

[Входные, выходные данные, а также ограничения приложения 8](#_Toc103866429)

[1.3. Описание и выбор средств разработки 8](#_Toc103866430)

[Выбор инструмента для создания мобильного приложения 8](#_Toc103866431)

[Выбор базы данных для мобильного приложения 13](#_Toc103866432)

[2. Создание базы данных 15](#_Toc103866433)

[2.1. Таблица work\_group\_name 16](#_Toc103866434)

[2.2. Таблица employee 16](#_Toc103866435)

[2.3. Таблица auth\_table 17](#_Toc103866436)

[2.4. Таблица time\_table 17](#_Toc103866437)

[2.5. Таблица marker\_table 18](#_Toc103866438)

[2.6. Таблица watcher\_table 19](#_Toc103866439)

[2.7. Таблица watcher\_table\_skip 20](#_Toc103866440)

[2.8. Функция count\_skip 21](#_Toc103866441)

[2.9. Функция skip\_time 21](#_Toc103866442)

[2.10. Представление watcher\_all 21](#_Toc103866443)

[2.11. Представление admin\_employeers 22](#_Toc103866444)

[2.12. Процедура proc\_skipt0 23](#_Toc103866445)

[2.13. События, связанные с расписанием 23](#_Toc103866446)

[3. Разработка мобильного приложения 24](#_Toc103866447)

[3.1. Диаграмма классов 24](#_Toc103866448)

[Класс Database 24](#_Toc103866449)

[Класс employee 26](#_Toc103866450)

[Класс Shedule 27](#_Toc103866451)

[Класс Marker 29](#_Toc103866452)

[Класс Group 30](#_Toc103866453)

[Класс Worker\_skip 31](#_Toc103866454)

[Класс WatchWorker 32](#_Toc103866455)

[3.2. Разработка пользовательского интерфейса 32](#_Toc103866456)

[Блок авторизации 32](#_Toc103866457)

[Представления «Смотритель» 33](#_Toc103866458)

[Представления «Администратора» 34](#_Toc103866459)

[4. Реализация и тестирование программы 39](#_Toc103866460)

[4.1. Тестирование программы 39](#_Toc103866461)

[Заключение 40](#_Toc103866462)

[Список литературы 41](#_Toc103866463)

[Приложение А. Листинг программы. 42](#_Toc103866464)

# ВВЕДЕНИЕ

Каждый работодатель желает, чтобы его работники были продуктивными и выполняли свою работу к сроку. Для достижения этих целей есть множество способов. Один из них установка определенного расписания работы для работников, но в силу непредвиденных обстоятельств и человеческого фактора, каждый человек может опоздать и не прийти на определенную встречу к сроку.

До сих пор на некоторых местах работы по приходу на работу ведется журнал о прибытии работника на рабочее время, а также в нем фиксируется время его ухода с рабочего места, но с приходом цифровой эры данный метод контроля нахождения работника на рабочем месте модифицируется. К примеру, при проходе на работу устанавливают КПП, чтобы пройти через него, необходимо воспользоваться пластиковой карточкой или любым другим предметом с NFC-меткой. Данный метод позволяет автоматически идентифицировать человека, который собирается пройти на рабочее место и зафиксировать время его прибытия. Его плюсом является то, что от процесса записи в бумажный журнал ФИО работника и времени прибытия или ухода мы перешли к обычному прикладыванию пластиковой карты и ее последующему считыванию.

В свою же очередь, возможности современного телефона позволяют создать систему, которая будет уведомлять о пересечении определенных зон. Это все возможно благодаря технологии геопозиционирования или геолокации. Это весьма удобное решение, так как для современного человека телефон стал неотъемлемой частью его жизни и выход из дома без телефона представляется для человека трагедией, так как телефон обрел широкий функционал необходимый для комфортного существования в современном мире.

В работе предлагается использовать вместо пластиковых карт или других предметов, которые позволили бы идентифицировать человека и вести учет его посещаемости – его мобильный телефон.

Целью данной практики является закрепление, углубление и расширение теоретических знаний, умений и навыков, полученных в процессе теоретического обучения, обоснование и систематизация информационного материала, необходимого для написания выпускной квалификационной работы.  
Для выполнения цели практики были поставлены задачи:  
1.      Закрепление и систематизация теоретических знаний, полученных в процессе обучения;

2.      Закрепление опыта, навыков, развитие умений самостоятельной работы с нормативно-правовой, нормативно-методической и справочной литературой;  
3.      Сбор, обработка и подготовка материалов для выполнения выпускной квалификационной работы. За время прохождения практики необходимо изучить теоретический материал и подобрать литературу по следующим темам: БД MySQL, разработка мобильных приложений при помощи Xamarin.Forms, С# .NET. Необходимо разработать приложение на языке XAML и С#, необходимо подготовить материалы к расчетно-пояснительной записке выпускной квалификационной работы в соответствии с требованиями.  
В результате практики должно быть разработано приложение, позволяющее определять наличие работников на своих рабочих местах при помощи технологии геолокации.

# Задание

## Описание предметной области

Геолокация являлась неотъемлемой частью людей, которым требовалось преодолевать большие расстояния во все времена. Во время морских плаваний, дальних путешествий, в каждом из них было необходимо ориентироваться в пространстве. С давних времен ориентиром могли быть карты, проложенные тропы, а также указатели, размещение звезд на небе. В дальнейшем в ориентации помогал компас, в случае если не было видно небо. Спустя многие века технологии совершенствовались и военными США была запущен система навигации под названием GPS. Система GPS – это система спутниковой навигации, которая позволяет почти в любом уголке земли (кроме приполярных областей) и любой погоде определить текущее местоположение. Основной принцип использования системы — определение местоположения путём измерения моментов времени приёма синхронизированного сигнала от навигационных спутников антенной потребителя. У России и Европы есть аналоги данной системы, а именно ГЛОНАСС и GALILEO. Со временем стали появляться способы определения текущего местоположения благодаря вышкам сотового оператора, Wi-Fi сетей и прочие. Они действуют почти также как и вышеуказанная спутниковая система, только вместо спутников используются телефонные вышки и роутеры, положение которых известно. На основе этих данных можно получить точность текущего местоположение вплоть до десятков метров. Все было так вплоть до выхода Google maps 2.0, которые позволяли определять текущее местоположение средствами сервисов Google, что составляло конкуренцию для мобильных операторов, которые зарекомендовали себя как эффективный способ получения текущего местоположения.

Сейчас основными являются 3 технологии позволяющие определить текущее местоположение телефона в пространстве:

* Система GPS (или подобные ей ГЛОНАСС, GALILEO)
* Системы на основе вышек операторов сотовой связи
* Сервисы Google (начиная с Google maps 2.0)

### **Система GPS (или подобные ей** ГЛОНАСС, GALILEO**)**

Система GPS представляет из себя спутниковую систему навигации и для полноценной работы должна содержать по крайней мере 24 спутника. На момент 13 сентября 2021 года используется 32 спутника. Для работы системы GPS необходимо включить опцию «Местоположение».Для точного вычисления координат необходимо минимум 4 спутника — три передают данные о своем местоположении относительно земли и друг друга, четвертый фиксирует время прохождения сигнала от передатчика к приемнику. Вычисление местоположение телефона определяется процессором устройства и является весьма трудоемкой задачей. На текущий момент некоторые модели телефонов могут обрабатывать сигнал от 12 спутников одновременно. Чем больше одновременно может быть обработано спутниковых сигналов, тем более точные данные о местоположении получит пользователь.

Для полноценного определения местоположения также еще требуется данные с определенных датчиков:

Акселерометр приходит на помощь спутникам, показывая куда поворачивает телефон, с какой скоростью он движется.

Магнитометр помогает акселерометру понять, где север, чтобы сориентировать в пространстве. Чем больше всяких датчиков в гаджете, тем точнее будут определены координаты.

Датчик компаса помогает определить направление движения. Если он не настроен, точность позиционирования значительно снижается.

Для определения трёхмерных координат GPS-приёмнику нужно иметь четыре уравнения: «расстояние равно произведению скорости света на разность моментов приёма сигнала потребителем и момента его синхронного излучения от спутников»:

Здесь:

- радиус-вектор j-го спутника,

- момент времени приёма сигнала от j-го спутника по часам потребителя,

- неизвестный момент времени синхронного излучения сигнала всеми спутниками по часам потребителя,

– скорость света,

– неизвестный радиус-вектор потребителя.[1]

### **Геолокация на основе вышек операторов сотовой связи**

Всего мобильные операторы поддерживают множество методов поиска, которые не обязательно должны использовать технологию GPS или ГЛОНАСС. Основные методы поиска приведены ниже:

* **Cell of Origin** — простейший метод, позволяющий узнать местонахождение мобильного телефона по известному CellID (идентификатору вышки сотовой связи). В его плюсы можно отнести то, что для его работоспособности не требуются какое-то дополнительные возможности или оборудование для телефона, всего лишь достаточно установки программного комплекса и MLC (Mobile Location Center, центр мобильного позиционирования). Координаты вычисляются на известных данных, а именно на данных о расположении вышек сотовой связи. Точность данного метода зависит от количества сотовых вышек в округе. К примеру, в центре крупного города точность обычно составляет несколько сотен метров, в небольших городах и поселках точность составляет около километра. В сельской местности точность снижается до 35 км. В минусы данного способа можно отнести то, что при отсутствии сотовой связи узнать свое местоположение не предоставляется возможным.
* **TOA** (Time of Arrival, Оценка времени прибытия сигнала) — данный метод основан на алгоритме трилатерации. В данном методе измеряются и сравниваются интервалы времени прохождения сигнала от мобильного телефона до нескольких базовых станций, при этом данные станции должны быть оснащены LMU (Location Measurement Unit, блок определения местоположения). С этими данными управляющий компьютер сети, используя алгоритм трилатерации, вычисляет по разности времени поступления сигналов местоположение передатчика. Затем информацию о местонахождении передатчика отправляют клиенту или на сервер для дальнейшей обработки данных и использовании их в различных сервисах.
* **OTD** (Observed Time Difference, Наблюдаемая разность времени прибытия сигнала) — основан на измерении и сравнении интервалов времени прохождения сигналов от нескольких базовых станций до мобильного телефона абонента. Требует модернизации сетевого оборудования, однако такая модернизация значительно дешевле TOA. Управляющий контроллер мобильного телефона измеряет время прохождения сигнала от нескольких базовых станций, одна из которых оснащена блоком LMU. Для получения информации о своем местоположении абонент совершает звонок, при котором его телефон до установки речевого соединения посылает специальное сигнальное сообщение, MLC производит необходимые вычисления для расчета местоположения, после чего пакет данных с координатами местонахождения абонента пересылается на сотовый телефон.

## Постановка задачи

Целью данной преддипломной практической работы является разработка мобильного приложения и БД, которая будет находится на удаленном сервере. Данное мобильное приложение необходимо для контроля работников компании, которое позволяет отслеживать время прибытия и пребывания персонала на рабочем месте.

Данное мобильное приложение будет определять время прибытия и ухода работника из рабочей области в фоновом режиме. Данные и прибытии, и уходе приложение будет передавать в БД. Также к этой БД будет иметь доступ определенный круг работников под названием «*Смотрители*». Им будут выводится все пропуски и прогулы других работников, а также они будут в праве удалять записи о пропусках из базы данных, в случае если работник пропустил работу по уважительной причине. Также будет предусмотрен тип работников «*Администраторы*», которые сумеют создавать, редактировать и удалять уже существующие рабочие группы, работников, указывать рабочие области для определенных групп. Также как было указано выше, необходимо предусмотреть взаимодействие с работниками компании, за которыми будут следить. Это взаимодействие будет связано с показом информации о пользователе, его пропусках и времени, когда они были совершенны. Также будет показываться карта с текущим местоположением пользователя и границы рабочей области. В случае покидания которой в рабочее время будет показываться уведомление о необходимости вернуться обратно.

Необходимо реализовать авторизацию в мобильном приложении и по итогу прохождения которой будет определяться личность пользователя, его тип работника (Администратор/Смотритель/Работник) и высвечиваться необходимый пользовательский интерфейс.

**В текущей работе планируется разработать только БД, модуль авторизации пользователей, а также пользовательский интерфейс для Смотрителя и Администратора.**

**Пользовательский интерфейс для Работников будет разработан и представлен в выпускной бакалаврской работе.**

### Входные, выходные данные, а также ограничения приложения

## Описание и выбор средств разработки

### Выбор инструмента для создания мобильного приложения

Для разработки мобильного приложения необходимо выбрать для какого из ОС для смартфона будет создана программа или же написать кроссплатформенное приложение.

Для начала необходимо проанализировать доли мобильных ОС на рынок смартфонов, так как наше приложение будет устанавливаться на телефоны работников компании и в наилучшем случае необходимо покрыть наибольшее количество ОС для телефонов, чтобы отслеживать время пребывания работников на своем рабочем месте.

На данный момент на рынке сейчас присутствует 3 основных вида ОС для мобильных телефонов:

* Android
* IOS
* Windows Phone (Поддержка прекращена в 2019 году)

Разделение же между рынка между данными ОС для телефонов приведены на рис. 1.

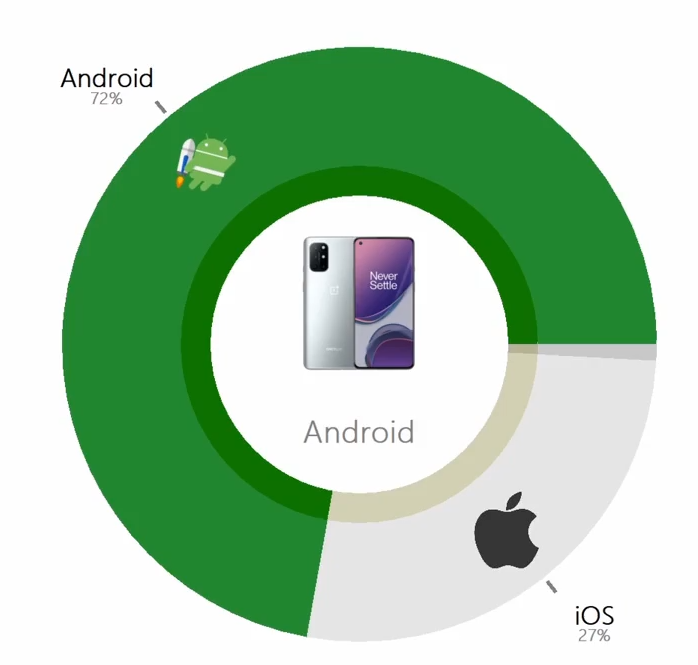


Рис. 1. Доля рынка смартфонов на разных устройствах.

Данная диаграмма действительна для июля 2021 года.

Как видно из диаграммы на данный момент есть две основных ОС: Android и IOS, которые в сумме занимают 99% рынка.

То есть для достижения наших целей необходимо покрыть эту группу устройств. В данном случае есть 2 выхода:

1. Написать отдельно мобильное приложения для каждой из указанных ОС. При этом в следствии ограниченного числа лиц, создающих данные приложение и рекомендаций определенных языков программирования для создания собственных приложений (native applications). Для IOS Apple рекомендует использовать язык Swift. Для Android Google советует использовать Kotlin (хотя для создания собственных (native) приложений на андроид может также подойти язык программирования Java). Итого имеем, что необходимо дважды создать одно и тоже мобильно приложение с использованием разных языков программирования.
2. Использовать кроссплатформенные инструменты, которые позволят создать приложение, которое будет работать на обоих ОС для телефонов. Выбирая этот метод, мы не столкнёмся с трудностями из пункта 1, так как нам придется разработать приложение 1 раз и на 1 языке программирования.

Большинство больших компаний (Meta, Google, Microsoft) решили использовать второй метод и создали для этих целей решения для кроссплатформенной разработки.

Meta (Экстремистская в РФ организация) – разработала React Native, на котором работают мобильные приложения facebook, Instagram, также на нем написано несколько отечественных программ, к примеру приложение для доступа к электронной медицинской карте. Для того чтобы работать на нем необходимо знать язык программирования JavaScript и TypeScript.

Google разработала Flutter, на котором работает приложение Alibaba, Tencent. В данный момент он является наиболее популярным инструментом для кроссплатформенной разработки мобильных приложений, но для разработки на нем необходимо знать специализированный, но легкий в освоении язык программирования Dart, он был создан и продвигается компанией Google.

Microsoft, которые приобрели Xamarin и стали ее развивать. Оно также позволяет создавать и писать кроссплатформенные мобильные приложения Kellog’s, Slack, Bosh. Для разработки с использованием Xamarin.Forms необходимо знать язык программирования C# и желательно быть знакомым с языком разметки XAML.

Так как автор данной работы имеет опыт работы с языком C# и работал с языками разметки на подобии XAML, то выбор сделан в сторону Xamarin.

#### Xamarin

Xamarin является платформой с открытым исходным кодом, которую используют для создания современных, а также высокопроизводительных приложений для таких операционных систем как iOS, Android и Windows с .NET. Xamarin является неким уровнем абстракции, благодаря которому обеспечивается управление между общей кодовой базой и кодом для определенной платформы. Также стоит отметить, что Xamarin выполняется в управляемой среде на основе .NET, благодаря чему реализовано большое количество возможностей, таких как выделение памяти и сборка мусора.

Одним из главных преимуществ Xamarin можно назвать то, что в среднем не менее 80 % кода приложения может использоваться без изменений на различных платформах, то есть быть общим кодом, который был написан разработчиком 1 раз, что значительно удешевляет разработку мобильного приложения, а также при всем прочем приложение будет работать с скоростью, близкой с скоростью, приложений написанных для определенных платформ. Еще данное приложение будет иметь такое же оформление и поведение, как и у платформ, на которых они будут использоваться, то есть пользователю будет крайне комфортно использовать приложение, написанное с использованием Xamarin и его отличия от других приложений будут минимальны. [2]

#### Xamarin.Forms

Xamarin.Forms предоставляет из себя API для использования и создания элементов пользовательского интерфейса на различных платформах, на котором работает Xamarin. То есть при помощи Xamarin.Forms можно создать и использовать пользовательский интерфейс под 3 различные платформы, описать взаимодействие с данным интерфейсом. Создание пользовательского интерфейса в Xamarin.Forms доступно 2 путями. Разработчики могут использовать и создавать пользовательские интерфейсы в XAML или описать его в программной части с использованием C#. Данная возможность открывает дополнительные возможности и в определенных случаях упрощает процесс создания пользовательского интерфейса. [2]

Принцип работы Xamarin и Xamarin.Forms.

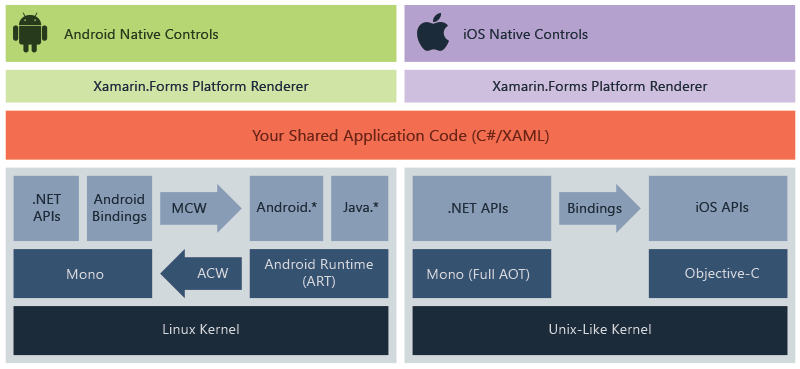


Рис. 2 Схема общей архитектуры кроссплатформенного приложения на Xamarin.Forms

#### XAML

XAML является языком разметки для eXtensible приложений, был разработан компанией Microsoft для замены программного кода, описывающего пользовательский интерфейс. В его основные задачи входит создание и инициализацию объектов интерфейса и применение организации к объектам для иерархии «родитель-потомок». В его основе лежит язык XML.

Язык XAML изначально применялся в .NET Framework, но с течением времени, его стали использовать в Silverlight, UWP, а также в WPF.

Также XAML стали применять в Xamarin.Forms. При этом его использование необязательно. Без него можно вполне реального создавать мобильные приложения используя Xamarin.Forms. Но XAML приносит ряд преимуществ, по сравнению с программным кодом, такие как краткость и читабельность. Еще одним преимуществом можно назвать, крайняя легкость в использовании архитектурного паттерна для приложений MVVM, где благодаря XAML разработчик расписывает View, а при помощи привязки данных (Binding) создается связь с ViewModel, который уже расписан при помощи кода.

XAML в Xamarin.Forms имеет большее количество инструментов и возможностей для определения пользовательского интерфейса при помощи уже готовых страниц, представлений и макетов, а также допускается создания собственных пользовательских классов или переопределение функционала уже существующих классов.

Принцип использования XAML-файлов заключен в следующем: Все XAML-файлы анализируются при сборке проекта для того, чтобы найти все именованные объекты. Затем XAML-файлы используются при запуске мобильного приложения, для того чтобы создать, а также проинициализировать все необходимые объекты, а также связи между ними и программным кодом, связанным со страницей, представлением или макетом.

Теперь следует осветить преимущества и недостатки XAML. Язык разметки XAML имеет следующие преимущества:

* Использование языка XAML облегчает создание пользовательского интерфейса для разработчика, улучшает его читаемость и делает гораздо лаконичнее, по сравнению с программным кодом.
* Большой набор уже созданных инструментов а, также возможность создавать собственные или изменять уже существующие классы для создания пользовательского интерфейса
* Возможность применения иерархии «родитель-потомок», которая унаследована из XML.
* Использование архитектурного паттерна MVVM и привязки данных

В минусы XAML можно отнести основные недостатки языков разметки:

* Нельзя использовать циклы, для повторной обработки
* Нельзя использовать условные переходы
* Нельзя вызывать методы

Данные минусы относятся только к пользовательскому интерфейсу, который описан только при помощи языка XAML. В случае использования вместе с XAML программный код, который будет описывать поведение или некоторые элементы интерфейса, то данные ограничения можно будет обойти.

Также к одному из минусов можно отнести отсутствие визуального конструктора, который визуализировал бы код, написанный на XAML в режиме реального времени. По итогу, для того чтобы разработчик увидел описанный им пользовательский интерфейс необходимо, чтобы он скомпилировал приложение и запустил его на реальном устройстве или эмуляторе, но благодаря добавлении функции «Горячей перезагрузки» в Visual Studio 2019 или Visual Studio для Mac, можно менять прямо на ходу содержимое XAML-файла уже скомпилированного приложения и видеть все внесенные изменения в режиме реального времени. [2]

### Выбор базы данных для мобильного приложения

Для полноценного функционирования мобильного приложения необходимо создать сервер с БД, к которому будет обращаться телефон пользователя для получения актуальной информации и для отправки данных для хранения и/или обработки.

Так как это приложение нужно для отслеживания работников, оно не подразумевает хранения больших объемов (не более 2 ГБ) данных и их анализ. Также записи, которые будет хранить БД, будут иметь одинаковую структуру. Также стоит отметить, что данная БД не будет сильно расширяться или уменьшаться. Из всего вышеперечисленного можно сделать вывод, что нам больше подойдет реляционная СУБД, нежели не реляционная СУБД.

Далее следует саму реляционную БД, ниже перечислены самые популярные реляционные БД:

* PostgreSQL
* Oracle
* Microsoft SQL Server
* MySQL

Рассмотрим преимущества каждой из них и выберем наиболее подходящее для данного проекта.

PostgreSQL – бесплатное программное обеспечение с открытым исходным кодом. Имеет нормальную производительность и легкую расширяемость. Также поддерживает многоверсионность, что необходимо в работе нашего приложения. Среди минусов стоит отметить, что PostgreSQL не так популярна, при использовании несколькими клиентами требует для своей работоспособности большое количество памяти. В основном работает на Linux/UNIX системах.

Oracle – Предназначена для хранения больших объемов данных, позволяет подключаться большему количеству клиентов одновременно и выполнять различные запросы. На ней держаться самые большие базы данных в мире. Её используют в крупных компаниях, для хранения огромных массивов данных, к которым обращается большое количество клиентов. Так как наше мобильное приложение не имеет больших требований к объемам данных и не требует постоянного доступа к данным для огромного количества клиентов, то можно сказать, что данные преимущества нам не нужны.

Microsoft SQL Server – Данное программное обеспечение является бесплатным только в образовательных целях или в целях разработки. В коммерческих целях его использование крайне затратно по сравнению с аналогами. Данная СУБД размещается только на серверах под управлением Windows, которых по статистике меньшинство, то есть будет крайне ограниченный выбор серверов, для размещения на них разработанной БД на основе данного СУБД.

MySQL – является одним из самых популярных и распространённых СУБД. Весьма прост в освоении. Имеет высокую производительность и поддерживает большинство возможностей, которые предоставляют другие СУБД (возможности, предоставляемые этой СУБД полностью хватает для создания БД для нашего мобильного приложения). Имеет поддержку на всех основных платформах.

В следствии всех перечисленных СУБД, их преимуществ, хороших сторон, а также минусов было выбрано в качестве СУБД MySQL.

Для создания БД и проверки работоспособности БД будет использоваться следующее программное обеспечение:

* Apache HTTP Server версии 2.4
* PHP версии 8.0.3
* MySQL версии 8.0
* phpMyAdmin версии 5.10

В дальнейшем планируется выгрузка созданной БД в базу данных Azure для MySQL и ее настройка.

# Создание базы данных

Для полноценной работы приложения нам необходимо сохранять и читать данные, которые содержат важную информацию. Для этой цели будет использоваться база данных. Также при ее разработке необходимо ее спроектировать таким образом, чтобы все возможные вычисления, которые возможны на стороне сервера, выполнялись там, а также максимально уменьшить количество обращений к серверу, чтобы снизить нагрузку на каналы связи, которые соединяют мобильные телефоны нескольких клиентов с базой данных.

Объединив все данные, необходимые для мобильного приложения и проведя нормализацию до 3 нормальной формы было получено 7 таблиц, каждая со своими полями и с связами между друг другом. Некоторые таблицы предусматривают добавление триггеров. Также для получения данных из нескольких таблиц будет использоваться 2 представления. Данные таблицы и представления приведены ниже на рис. 2

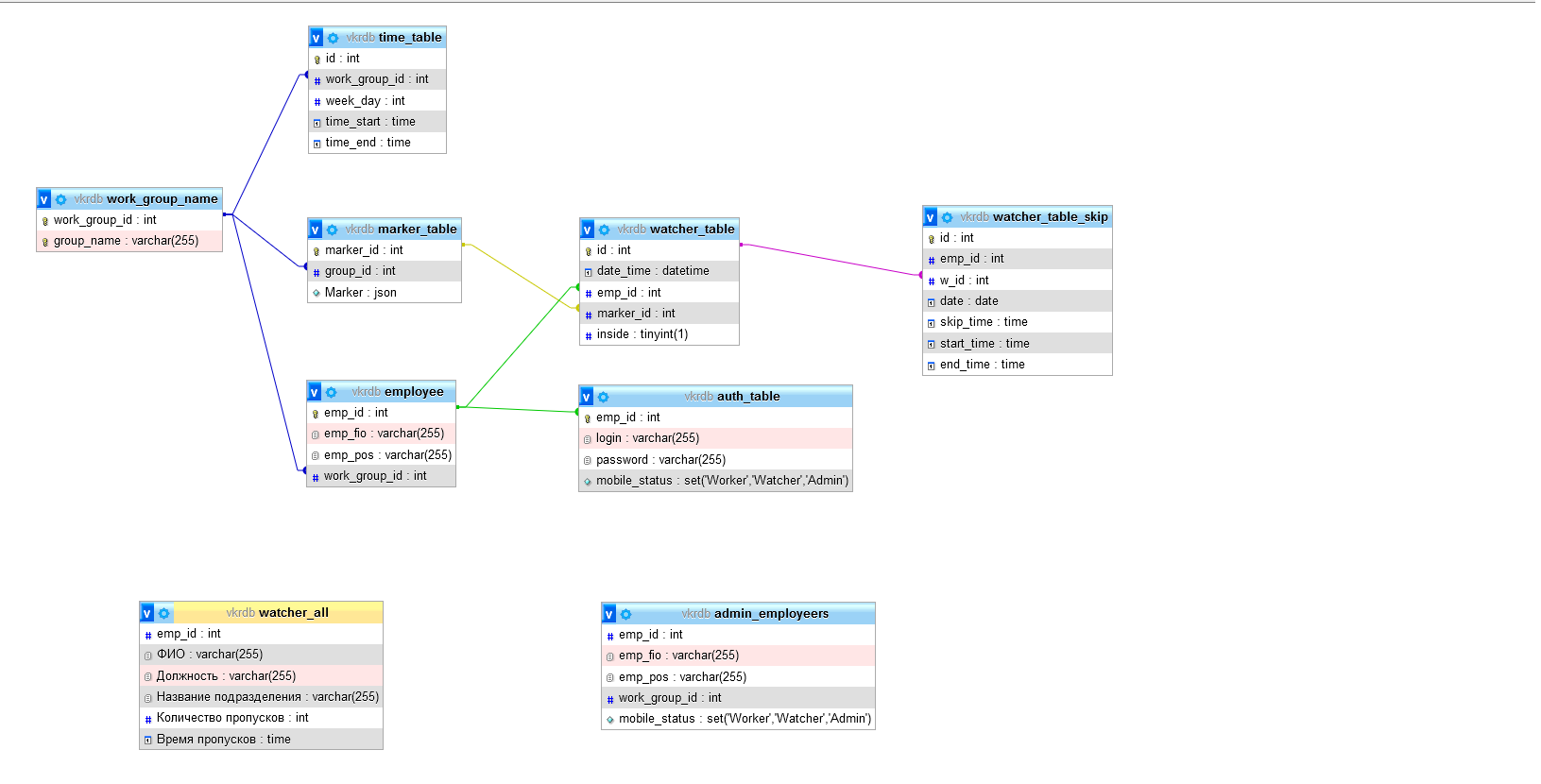


Рис. 2 Диаграмма таблиц и представлений

Для доступа к определенным таблицам будет создано 3 вида пользователей со своими разрешениями: Администратор, Смотритель, Рабочий. Их привилегии и уровни доступа будут описаны для каждой таблицы отдельно.

Для обработки данных и подсчета определенных результатов будет использоваться 2 функции и 1 процедура, а также будет использоваться планировщик событий и сами события, которые будут рассмотрены ниже, при описании каждой из таблиц.

## Таблица work\_group\_name

Данная таблица хранит в себе порядковый номер рабочей группы, в которой будут работники компании (среди которых могут и «Смотрители», и «Администраторы»), а также название рабочей группы. Прямой доступ к этой таблице имеет только класс пользователей «Администратор», который может читать, добавлять, изменять, удалять записи из таблицы.

Данная таблица имеет 2 поля:

* work\_group\_id типа int
* group\_name типа varchar(255)

Поле work\_group\_id имеет ограничение **первичного ключа** и обладает свойством AUTO\_INCRIMENT, что позволяет нам при добавлении записи в таблицу не указывать явно номер рабочей группы, а также помогает нам избежать ситуации, когда значения данного поля одинаковые, что не является возможным из-за ограничения **первичного ключа**. В случае, если пользователь явно указывает номер рабочей группы, и он не является уникальным, возникает ошибка и БД отправляет нам уведомление, о том, что запись с данным номером рабочей группы уже существует.

## Таблица employee

Данная таблица содержит информацию о работниках в компании, а именно она содержит уникальный идентификатор работника, его ФИО, должность, а также номер рабочей группы, в которой он работает. К данной таблице напрямую имеет доступ только класс пользователей «Администратор», который может читать, добавлять, изменять, удалять записи таблицы.

Данная таблица имеет 4 поля:

* emp \_id типа int
* emp\_fio типа varchar(255)
* emp\_pos типа varchar(255)
* work\_group\_id типа int

Поле emp\_id является первичным ключом с свойством AUTO\_INCRIMENT и хранит в себе уникальный идентификатор работника.

Поле emp\_fio хранит в себе ФИО работника.

Поле emp\_fio хранит в себе должность работника.

Поле work\_group\_id хранит в себе рабочую группу пользователя и является внешним ключом к полю work\_group\_id в таблице **work\_group\_name** и имеет каскадную зависимость на изменение и удаление. Это означает, что в случае изменения/удаления записи с полем work\_group\_id в таблице **work\_group\_name** все записи, которые имели такое же значение поля с внешним ключом и каскадным ограничением в данной таблице (employee) будут автоматически изменены/удалены.

К примеру, при удалении записи о рабочей группе с порядковым номером 2 из таблицы **work\_group\_name,** все записи, которые содержали в качестве значения поля work\_group\_id число равное 2-м в таблице e**mployee** будут также удалены.

## Таблица auth\_table

Данная таблица содержит информацию необходимую для авторизации в приложении, а именно уникальный идентификатор пользователя логин, пароль и уровень привилегий пользователя. Данная таблица будет доступна для класса «Администратор», он имеет права добавлять, изменять и удалять записи из таблицы, но не читать такие конфиденциальные данные, как логин и пароль.

Данная таблица имеет 4 поля:

* emp\_id типа int
* login типа varchar(255)
* password типа varchar(255)
* mobile\_status типа set со значениями «Worker», «Watcher», «Admin»

Поле emp\_id хранит в себе уникальный идентификатор пользователя. Данное поля имеет ограничения первичного ключа, а также внешнего ключа, который ссылается на поле emp\_id в таблице employee.

Поле login хранит в себе логин, который необходимо ввести пользователю, для прохождения аутентификации.

Поле password хранит в себе пароль, который необходимо ввести пользователю, для прохождения аутентификации.

Поле mobile\_status хранит в себе значения уровня доступа для пользователя.

## Таблица time\_table

Данная таблица содержит информацию о расписании рабочих групп в компании. В ее состав входят поля, которые содержат информацию, о рабочей группе, о дне недели, времени начала работы и времени окончания работы. Данная таблица будет доступна для чтения классам пользователей «Работник» и «Администратор». Также «Администратор» имеет права добавлять, изменять и удалять записи из таблицы.

Данная таблица имеет 5 поля:

* id типа int
* work\_group\_id типа int
* week\_day типа int
* time\_start типа time
* time\_end типа time

Поле id является первичным ключом с свойством AUTO\_INCRIMENT и хранит в себе уникальный идентификатор записи таблицы.

Поле work\_group\_id хранит в себе номер рабочей группы.

Поле week\_day хранит в себе день недели, для которого указаны часы работы. Диапазон значений для данного поля равен от 0 до 6. Каждому значению соответствует свой день недели.

0 – воскресение

1 – понедельник

2 – вторник

3 – среда

4 – четверг

5 – пятница

6 - суббота

Поле time\_start хранит в себе время начала работы для данной рабочей в данный день недели.

Поле time\_end хранит в себе время окончания работы для данной рабочей в данный день недели.

Ниже на рис. 3 представлена запись о том, что для рабочей группы 1 в понедельник рабочими часами является промежуток времени с 10:00:00 до 18:00:00



Рис. 3 Пример записи в таблице time\_table

## Таблица marker\_table

Данная таблица содержит информацию о рабочих зонах для рабочих групп. В ее состав входят поля, которые содержат информацию о рабочей группе и соответственно сам маркер, который описывают рабочую зону. Данная таблица будет доступна для чтения классам пользователей «Работник» и «Администратор». «Администратор» имеет права добавлять, изменять и удалять записи из таблицы.

Данная таблица имеет 3 поля:

* marker\_id типа int
* group\_id типа int
* Marker типа json

Поле marker\_id является первичным ключом с свойством AUTO\_INCRIMENT и хранит в себе уникальный идентификатор записи таблицы.

Поле group\_id хранит в себе номер рабочей группы.

Поле Marker хранит в себе описание рабочей зоны в виде json строки.

Эта таблица нужна для того, чтобы работники получали актуальные рабочие зоны, которые будут создаваться/изменятся администраторами и загружаться с сервера им на телефон.

## Таблица watcher\_table

Данная таблица будет содержать информацию о пересечении работник границ рабочей зоны. На основе данных из этой таблицы будет получена информация о количестве времени, которое работник отсутствовал в рабочей зоне в рабочее время. В ее состав входят поля, которые содержат информацию о времени и дате пересечения рабочей области, об уникальном идентификаторе пользователя, который пересек рабочую зону, об уникальном идентификаторе пересеченного маркера и поле, сигнализирующее о выходе или входе в рабочую область. Данная таблица содержит информацию, которая будет только анализироваться и на основе уже проанализированной информации будут создаваться записи в таблицу watcher\_table\_skip.

Данная таблица имеет 5 поля:

* id типа int
* date\_time типа datetime
* emp\_id типа int
* marker\_id типа int
* inside типа Boolean

Поле id является первичным ключом с свойством AUTO\_INCRIMENT и хранит в себе уникальный идентификатор записи таблицы.

Поле date\_time хранит в себе дату и время пересечения рабочей области.

Поле emp\_id хранит в себе уникальный идентификатор пользователя, который пересек рабочую область, а также является внешним ключом, который ссылается на поле emp\_id таблицы employee.

Поле marker\_id хранит в себе уникальный идентификатор маркера. Данное поле имеет ограничения внешнего ключа, который ссылается на поле marker\_id таблицы marker\_table.

Поле inside определяет тип пересечения рабочей области работником. В случае если работник заходит в рабочую область, то поле inside = 1, в противном случае, когда работник покидает рабочую область, то inside = 0.

Также для таблицы watcher\_table был добавлен триггер на добавление записи в таблицу. Этот триггер работает, когда пользователь заходит внутрь рабочей области, затем сервером высчитывается время, которое рабочий пропустил из своей рабочей смены, указанный в таблице time\_table, для текущего дня недели. В случае, если время пропуска положительное и отличное от 0, то в таблицу watcher\_table\_skip добавляется запись, о пропущенном времени работника.

## Таблица watcher\_table\_skip

Данная таблица содержит информацию о пропусках работниками своего рабочего времени. Данная таблица содержит информацию о работнике, который отсутствовал на рабочем месте, о рабочей области, от которой пришло уведомление, дату пропуска, суммарное пропущенное рабочее время, время начала пропуска и время конца пропуска. Информация в данной таблицы будет использоваться для работы 2 функций, которые будут рассмотрены ниже, а также будет доступна для просмотра и изменения классом пользователя «Смотритель».

Данная таблица имеет 7 полей:

* id типа int
* emp\_id типа int
* w\_id типа int
* date типа date
* skip\_time типа time
* start\_time типа time
* end\_time типа time

Поле id является первичным ключом с свойством AUTO\_INCRIMENT и хранит в себе уникальный идентификатор записи таблицы.

Поле emp\_id хранит в себе уникальный идентификатор пользователя, для которого было зафиксировано отсутствие на рабочем месте, а также является внешним ключом, который ссылается на поле emp\_id таблицы employee.

Поле w\_id хранит в себе уникальный идентификатор, который является внешним ключом, ссылающимся на поле id таблицы watcher\_table.

Поле date хранит в себе дату, когда был совершен пропуск

Поле skip\_time хранит в себе время пропуска работников, которое было зафиксировано.

Поле start\_time хранит время, с которого работник отсутствовал на рабочем месте.

Поле end\_time хранит время, до которого работник отсутствовал на рабочем месте.

## Функция count\_skip

Данная функция анализирует таблицу watcher\_table\_skip и подсчитывает количество пропусков для определенного работника

Входные данные:

emp\_id (int) – Уникальный идентификатор работника

Выходные данные:

(int) – Количество пропусков данного работника

## Функция skip\_time

Данная функция анализирует таблицу watcher\_table\_skip и подсчитывает суммарное пропущенное время для определенного работника.

Входные данные:

emp\_id (int) – Уникальный идентификатор работника

Выходные данные:

(time) – Суммарное время пропусков для данного работника

## Представление watcher\_all

Данное представление необходимо для того, чтобы отображать классу пользователей «Смотритель» всю необходимую информацию о пользователях, количестве их пропусков и суммарном пропущенном времени пользователем.

Данное представление содержит 6 полей:

* emp\_id типа int
* ФИО типа varchar(255)
* Должность типа varchar(255)
* Название типа varchar(255)

подразделения

* Количество типа int

пропусков

* Время типа time

пропусков

Так как это представление, то все столбцы являются данными, которые уже имеются либо могут быть получены в данной БД.

Поля emp\_id, ФИО и Должность берут данные из таблицы employee из полей emp\_id, emp\_fio, emp\_pos соответственно.

Поле название подразделения содержит информацию о имени рабочей группы, в которой числится сотрудник. Оно получается следующим образом: из таблицы employee, помимо данных для выше указанных полей считывается уникальный идентификатор рабочей группы, к которой принадлежит работник. По этому идентификатору в таблице work\_group\_name определяется имя рабочей группы.

Поле «Количество пропусков» для получения данных о пропусках определенного работника использует функцию count\_skip. В качестве входных данных используется значение поля emp\_id.

Поле «Время пропусков» для получения данных о суммарном времени пропусков определенного работника использует функцию skip\_time. В качестве входных данных используется значение поля emp\_id.

## Представление admin\_employeers

Данное представление необходимо для наглядного представления данных для класса пользователей «Администратор». Данное представление содержит информацию о пользователе, а также предоставленным ему классом доступа в приложении.

Представление содержит 5 полей:

* emp \_id типа int
* emp\_fio типа varchar(255)
* emp\_pos типа varchar(255)
* work\_group\_id типа int
* mobile\_status типа Set

Первые четыре поля берутся из таблицы employee. Последнее поле берется из таблицы auth\_table.

## Процедура proc\_skipt0

Процедура proc\_skipt0 была создана для анализа таблицы watcher\_table и выявления пропусков работников из определенной рабочей группы. Данная процедура будет вызываться в конце рабочей смены при помощи событий, чтобы отследить отсутствующих работников и зафиксировать их пропуск. Данную процедуру планируется использоваться в событиях, которые будут создаваться в процессе добавления/изменения расписания рабочей группы и вызываться, как было уже указано выше, в конце рабочей смены.

Входные параметры

* w\_id (уникальный идентификатор рабочей группы) типа int
* time\_start (время начала рабочей смены) типа time
* time\_end (время окончания рабочей смены) типа time

## События, связанные с расписанием

При добавлении или изменении записей в таблице с расписанием рабочих групп будут создаваться повторяющиеся каждую неделю события, которые будут вызывать процедуру proc\_skipt0 в конце рабочей смены, определенного дня недели.

Данные события будут именоваться по определенному принципу, благодаря чему можно определить их принадлежность к рабочей группе и за какой день недели они отвечают. Название выглядит следующим образом wg\_<work\_group\_id>\_<week\_day>, где вместо <work\_group\_id> ставиться уникальный идентификатор рабочей группы, а вместо <week\_day> день недели, за который отвечает данное событие.

К примеру, событие wg\_1\_1 принадлежит к 1 рабочей группе и отвечает за понедельник.

# Разработка мобильного приложения

## Диаграмма классов

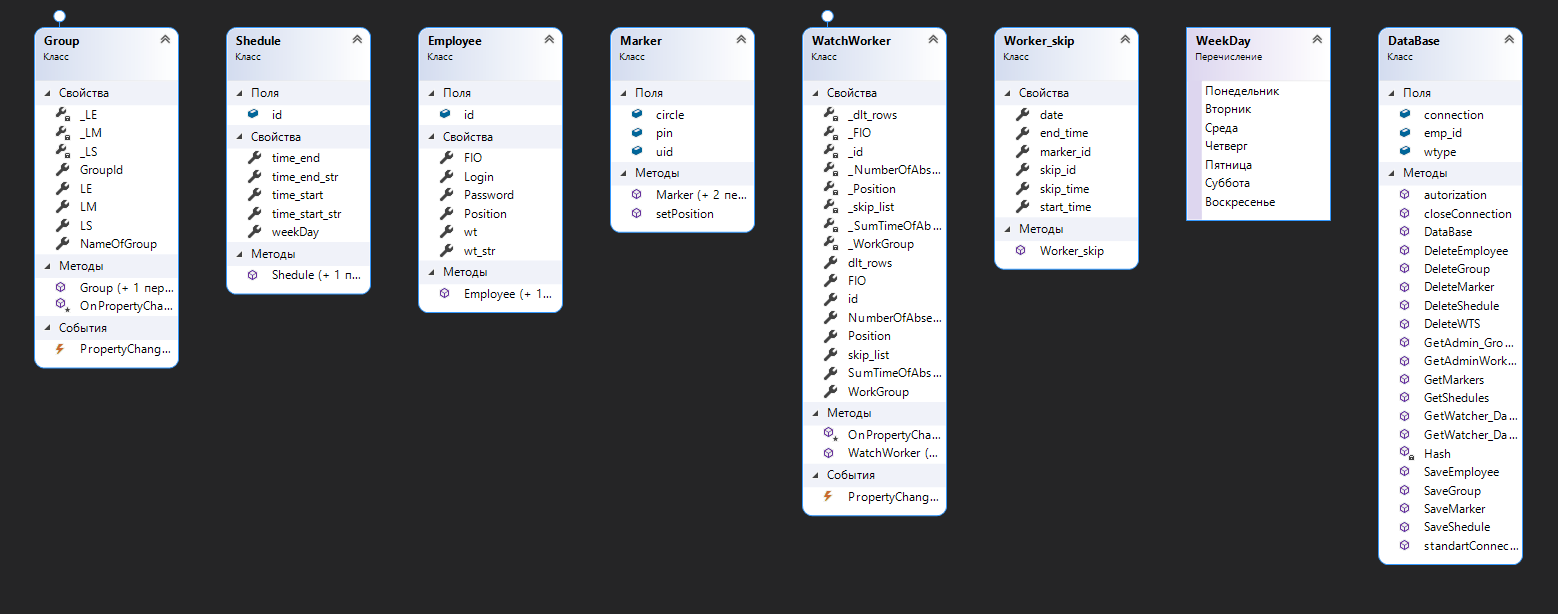


Рис. 4 Диаграмма пользовательских классов

На Рис.4 представлены все пользовательские классы, которые будут использоваться в данном приложении.

Рассмотрим каждый из этих классов по отдельности: его поля, методы их смысл и предназначение.

### Класс Database

Благодаря этому классу устанавливается соединение с сервером, на котором находится БД. Методы можно поделить на несколько типов: Методы для присоединения или отключения к БД, методы для авторизации в БД, методы для чтения/добавления/изменения/удаления записей из таблиц БД.

Поля класса DataBase:

* сonnection – данное поле является строкой подключения к БД
* emp\_id – данное поле хранит уникальный идентификатор пользователя, который работает с данным приложением
* wtype – Тип привилегий доступный пользователю с уникальным идентификатором emp\_id

Некоторые из методов класса DataBase возвращают результат на основе коллекций из пользовательских классов, поэтому такие методы будут рассматриваться в тех классах, коллекции которых они возвращают.

#### Метод standartconnection

Данный метод производит стандартное соединение с БД, если до его вызова соединение не было установлено. Он имеет два входных параметра: login (string) и password (string). Эти входные параметры имеют значения по умолчанию. В случае вызова этого метода без явного указания значений параметров, то произойдет соединение с БД с наименьшими привилегиями, то есть будут доступны лишь несколько таблиц только на чтение.

В качестве выходного параметра используется булевое значение, которое имеет значение «true», в случае если соединение было установлено, в противном случае возвращается результат «false».

#### Метод closeconnection

Этот метод используется для закрытия текущего соединения с БД.

Не имеет входных данных

Возвращает булевый результат, сигнализирующий об успешном закрытии соединения

#### Метод authorization

Данный метод, анализирует данные, которые ввел пользователь (логин и пароль) и в случае их достоверности выполняется процесс авторизации пользователя и присуждение ему уровня доступа, указанного в БД для данного аккаунта, а также заполнение таких полей как emp\_id и wtype.

Для получения доступа к БД будет использоваться метод standartconnection, если до этого соединение не было установлено.

Входными данными являются две строки login и password соответственно.

Выходным значением является булевая переменная, которая показывает правильность введенных данных сигнализирует об успешной процессе авторизации.

#### Метод Hash

Этот метод необходим для хэширования данных типа строки, которые могут передаваться по незащищенным каналам связи. Суть данного метода в сокрытии настоящих значений пароля и логина, введённых пользователем от злоумышленников, которые хотят завладеть этими данными.

Входные данные: строка input типа string

Выходные данные: преобразованная при помощи алгоритма SHA1 строка input

**Все остальные методы будут рассмотрены вместе с другими пользовательскими классами, так как используют данные пользовательские классы в качестве входных и выходных данных.**

### Класс employee

Объекты данного класса необходимы для получения, хранения и передачи информации о пользователе. Данный класс является моделью, которая используется в паттерне MVVM. В качестве информации, которые экземпляры данного класса получают и хранят являются такие свойства, как FIO (ФИО), Position (Должность) и wt (Уровень доступа в приложении). Взаимодействуя с представлением, пользователь класса «Администратор» может задать для определенного пользователя новый логин и пароль. Стоит отметить, что администратор не в праве узнать текущие конфиденциальные данные работников (логин и пароль), а может только задавать новые.

Единственным полем класса Employee является поле id (int), которое хранит в себе уникальный идентификатор пользователя.

Свойства класса Employee:

* FIO (string) - хранит в виде строки ФИО работника.
* Login (string) – хранит, заданный «Администратором» новый логин, который будет передан в БД и будет там сохранен.
* Password (string) – аналогично предыдущему полю, только содержит в себе пароль, а не логин.
* Position (string) – должность работника, записанная в виде строки.
* wt – является перечисляемым типом, имеет три значения: Admin, Watcher, Worker.
* wt\_str – является строковым эквивалентом свойства wt.

Все данные свойства могут быть заполнены и изменены классом пользователя «Администратор».

#### Метод GetAdminWorkers

Данный метод находится в классе DataBase. Он используется для получения коллекции Employee со всеми работника для указанного уникального идентификатора группы.

В качестве **входного** параметра данного метода выступает уникальный идентификатор группы, для которой необходимо получить коллекцию типа Employee, которая и является **выходными** данными данного метода.

Данный метод формирует запрос к БД, а потом из полученной информации формирует коллекцию Employee.

#### Метод SaveEmployee

Данный метод находится в классе DataBase. Суть данного метода в сохранении нового или фиксировании изменения внесенных в информацию о существующем работнике. Вся информация из данного метода фиксируется в БД.

В качестве входных параметров данный метод принимает объект класса Employee и уникальный идентификатор группы groupId (int).

Данный метод возвращает булевый результат, который сигнализирует об успешном выполнении метода. В случае возникновения непредвиденной ошибки, возвращается false.

#### Метод DeleteEmployee

Данный метод находится в классе DataBase, и он удаляет работника из указанной в экземпляре Employee рабочей группы БД.

В качестве входного параметра принимает экземпляр класса Employee.

Данный метод возвращает булевый результат, который сигнализирует об успешном выполнении метода. В случае возникновения непредвиденной ошибки, возвращается false.

### Класс Shedule

Экземпляры данного класса необходимы для хранения расписания рабочей группы. Shedule является моделью, которая будет использоваться в представлениях для таких классов пользователя, как «Работник» и «Администратор». Классу «Работник» будет доступно просмотр расписания своей рабочей группы, а классу «Администратор», помимо просмотра расписания всех рабочих групп, будет доступно добавления новых, изменения или удаление существующего расписания.

Единственным полем класса Shedule является поле id (int), которое хранит в себе уникальный идентификатор из таблицы time\_table.

Свойства класса Shedule:

* time\_start (TimeSpan) – хранит в себе время начала рабочей смены
* time\_start\_str (string) – является строковым эквивалентом свойства time\_start.
* time\_end (TimeSpan) - хранит в себе время конца рабочей смены
* time\_end\_str (string) – является строковым эквивалентом свойства time\_end.
* Weekday (enum weekday) – хранит в себе день недели, для которого указаны часы рабочей смены.

Все данные свойства могут быть заполнены и изменены классом пользователя «Администратор».

#### Метод GetShedules

Этот метод находится в классе DataBase. Его суть в получении коллекции расписания, по указанному уникальному идентификатору группы.

В качестве **входного** параметра данного метода выступает уникальный идентификатор группы, для которой необходимо получить коллекцию типа Shedules, которая и является **выходными** данными данного метода.

Данный метод формирует запрос к БД, а потом из полученной информации формирует коллекцию Shedules.

#### Метод SaveShedule

Данный метод находится в классе DataBase и используется для сохранения нового или фиксирования изменений в уже существующем расписании для рабочей группы. При добавлении нового рабочего дня, создается новое событие, которое будет вызываться в конце рабочего дня, для фиксирования отсутствия работников и подсчета пропущенного им в течении этого дня пропущенного времени, если таковое имеется. При изменении уже существующего расписания, будет удалено предыдущее событие и добавлено новое, обновленное.

В качестве входных параметров используется объект класса Shedule, а также уникальный идентификатор группы (int).

В качестве выходного значения служит булевая переменная, которая имеет значение «true» если сохранение прошло успешно и «false» в противном случае.

#### Метод DeleteShedule

Этот метод находится в классе DataBase и предназначен для удаления записи с расписанием из БД и удалением события, которое вызывается во время окончания рабочей смены.

В качестве входных параметров данный метод принимает объект класса Shedule, а также уникальный идентификатор группы.

### Класс Marker

Этот класс предназначается для хранения всей информации о рабочей области, в которой должны трудится сотрудники.

Она содержит следующие поля:

* circle – хранит в себе класс, который показывает границы рабочей области
* pin – сам маркер, который устанавливается в середину области. Он имеет собственный заголовок и пояснение к нему. Данное окно всплывает при нажатии на маркер.
* uid – уникальный идентификатор рабочей области. Он есть только у уже созданных и добавленных в БД рабочих областей.

С данным классом связанно 3 метода из класса DataBase:

#### Метод GetMarkers

Возвращает коллекцию объектов класса Marker. Данный метод нужен для получения всех рабочих зон для указанной рабочей группы. Данный метод отправляет запрос в БД и получает все данные для нужной рабочей группы из таблицы «marker\_table».

Входные данные: groupId (int) – UID рабочей группы, для которой необходимо получить коллекцию из Marker.

Выходные данные: Сама коллекция Marker.

#### Метод SaveMarker

Данный метод предназначен для сохранения маркера, который был создан или изменен «Администратором» при помощи работы с пользовательским интерфейсом. Созданный/Измененный экземпляр класса приводится к необходимому виду и загружается в БД в таблицу «marker\_table».

Входные данные:

* marker (Marker) – экземпляр класса Marker, который уже заполнен всей необходимой информацией.
* groupId (int) – UID рабочей группы, для которой необходимо сохранить эту рабочую область.

Выходные данные: булевая переменная, которая возвращает «true», если сохранение в БД прошло успешно, в противном случае метод возвращает «false».

#### Метод DeleteMarker

Данный метод предназначен для удаления маркера, который был выбран «Администратором».

Входные данные:

* marker (Marker) – экземпляр класса Marker, который уже заполнен всей необходимой информацией.

Выходные данные: булевая переменная, которая возвращает «true», если сохранение в БД прошло успешно, в противном случае метод возвращает «false».

### Класс Group

Данный класс предназначается для хранения всей информации про рабочую группу. А именно, в данном класс содержит следующие свойства:

* GroupId (int) - UID группы
* LE (коллекция Employee)
* LS (коллекция Shedule)
* NameOfGroup (string) – название рабочей группы.

Также данный класс имеет событие INotifyPropertyChanged и обработчик события OnPropertyChanged, которое вызывается при изменении свойств данного класса. Это все используется для поддержания паттерна MVVM.

С данным классом связано 3 метода из класса DataBase: GetAdmin\_Group\_Data, SaveGroup, DeleteGroup.

#### Метод GetAdmin\_Group\_Data

Данный метод используется для получения коллекции групп, которые будут отображены перед пользователем класса «Администратор».

Не имеет входных параметров.

Возвращает коллекцию Group.

#### Метод SaveGroup

Данный метод используется для создания нового или изменения уже существующей группы. В случае изменения, все изменения в названии рабочей группы будут происходить в каскадном порядке из-за внешних ключей со свойством «cascade».

В качестве входных параметром принимает название рабочей группы (string) и UID группы (int), которое имеет значение по умолчанию = -1, которое сигнализирует о добавлении новой рабочей группы. В случае изменения информации необходимо передать UID уже существующей рабочей группы.

Возвращает булевую переменную, которая имеет значение «true», в случае, когда сохранение прошло успешно и «false» если в процессе выполнения метода произошли ошибки.

#### Метод DeleteGroup

Данный метод используется для удаления рабочей группы.

В качестве входного параметра принимает UID группы.

Возвращает булевую переменную, которая имеет значение «true», в случае, когда удаление прошло успешно и «false» если в процессе выполнения метода произошли ошибки.

### Класс Worker\_skip

Данный класс предназначен для хранения данных о пропуске работника. В его свойства входят:

* skip\_id (int) – UID пропуска
* marker\_id (int) – UID рабочей области
* date (string) – дата пропуска
* start\_time (string) – время начала пропуска
* end\_time (string) – время окончания пропуска
* skip\_time (string) – само время пропуска

#### Метод GetWatcher\_Data\_Person

Класс Worker\_skip используется в методе GetWatcher\_Data\_Person, который возвращает коллекцию всех пропусков определенного работника. Данный метод находится в классе DataBase.

Входной переменной является UID работника, коллекцию пропусков которого мы хотим получить.

Выходным параметром является коллекция объектов Worker\_skip.

#### Метод DeleteWTS

Данный метод из класса DataBase, используется для удаления записи о пропуске из БД. Вызов этого метода доступно только классу пользователей «Смотритель» и «Администратор». Данный метод вызывается при нажатии на представлении на данные объекта класса Worker\_skip.

Входной переменной является UID пропуска.

Выходным параметром является булевая переменная, которая в случае успеха метода имеет значение «true», в противном случае «false».

### Класс WatchWorker

Данный класс содержит общую информацию о пользователе и его пропусках. Он предназначен для класса пользователей типа «Смотритель» и «Работник». Он содержит следующие свойства:

* id (int) – UID работника
* dlt\_rows – переменная которая определяет класс пользователя. В случае если приложением пользуется «Смотритель», то данная переменная позволяет выбирать и удалять пропуски, если они были совершены по уважительной причине или были отработаны. В случае если приложением пользуется «Работник», то он не имеет права удалять пропуски.
* FIO (string) – ФИО работника
* Position (string) – должность работника
* skip\_list (коллекция Worker\_skip) – коллекция всех пропусков работника
* SumTimeOfAbsence (TimeSpan) – суммарное время всех пропусков
* NumberOfAbsence (int) – количество пропусков данного работника
* WorkGroup (string) – наименование рабочей группы

#### Метод GetWatcher\_Data

Данный метод находится в классе DataBase. Этот метод формирует коллекцию из объектов класса WatchWorker, которая содержит всех пользователей и показывает всю необходимую информацию. Эта коллекция используется для представления «Смотрителя».

Входные данные отсутствуют.

Выходным параметром является коллекция WatchWorker.

## Разработка пользовательского интерфейса

### Блок авторизации

Данный блок должен иметь 2 текстовых поля, для заполнения логина и пароля, а также кнопку для входа в систему. Его внешний вид представлен на Рис. 5.

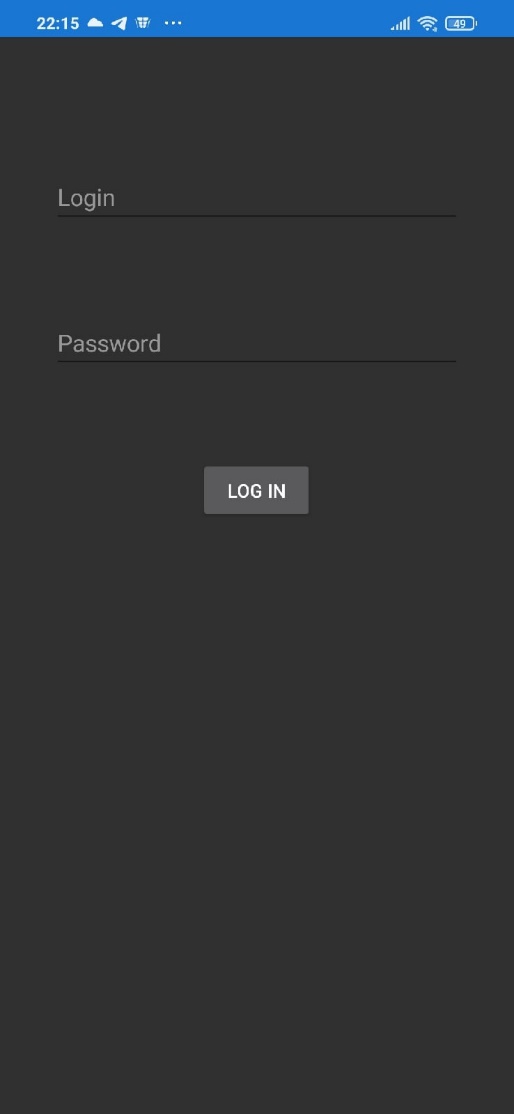
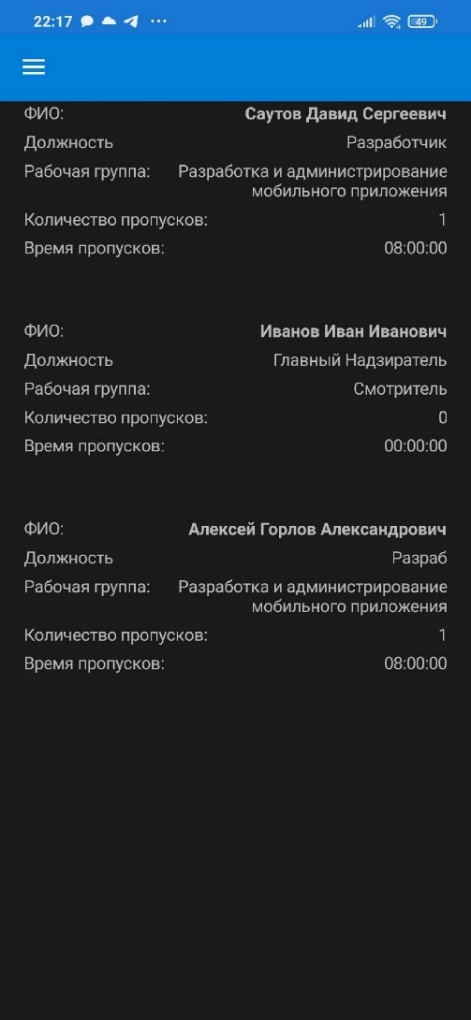
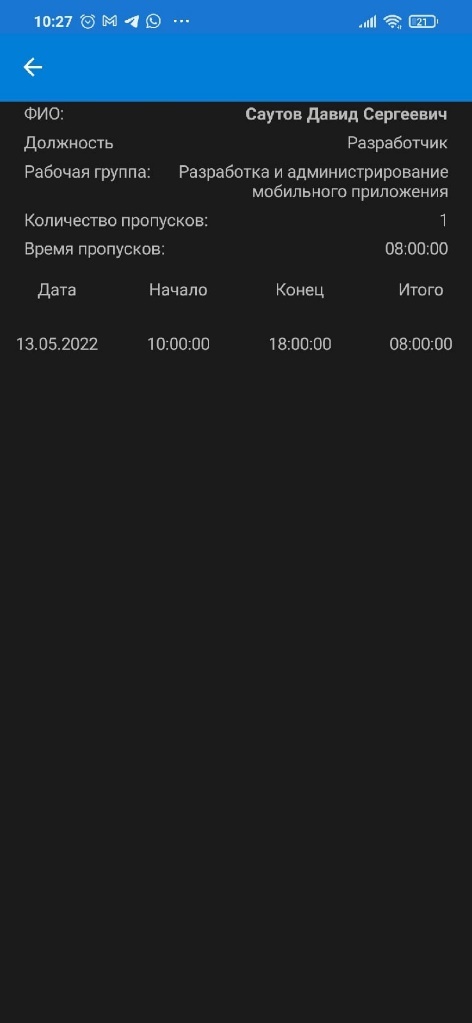
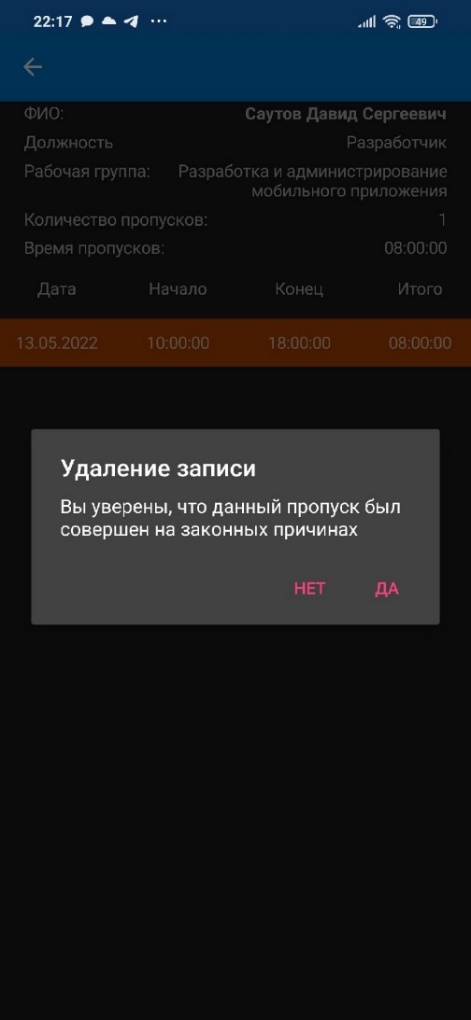


Рис. Блок авторизации

### Представления «Смотритель»

Смотрителю предоставлен список всех работников и информация об их пропусках, это представление видно на рис. 6. Также смотрителю предоставлена возможность удалять пропуски работников, если они были сделаны по уважительной причине Рис 6в.

а) б) в)

Рис 6. Представление для смотрителя. На рис. 6а видно представление со списком работников с их пропусками. На рис. 6б видны все пропуски для определенного работника. На рис. 6в показан процесс удаления пропуска.

### Представления «Администратора»

Для более простой навигации у администратора присутствует боковая панель навигации, а также верхняя панель с вкладками, на которой он может переходить между страницами, боковая панель представлена на рис. 8.

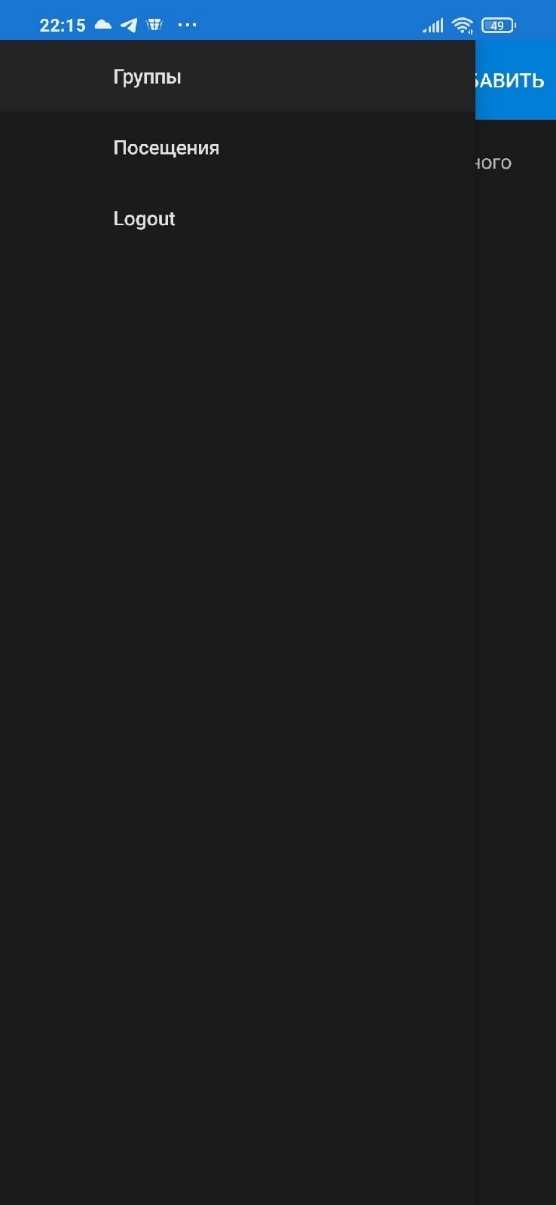
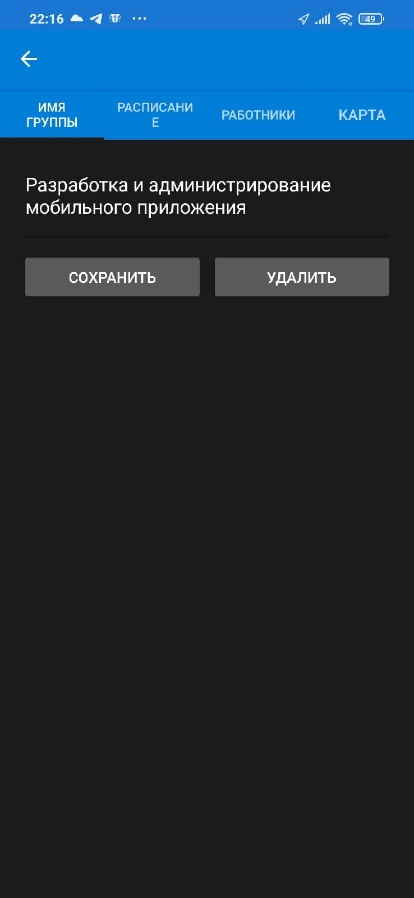
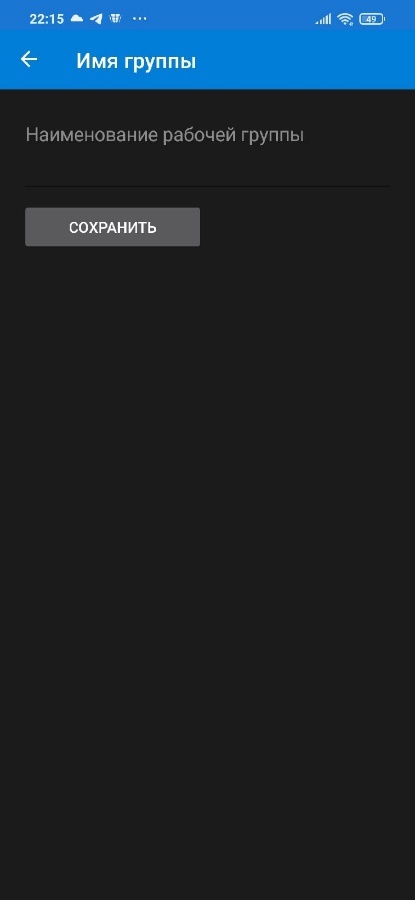


Рис. 7 Боковая панель администратора

При нажатии на раздел «Посещения» пользователь с привилегиями класса «Администратор» будет переведен к разделам работы с пропусками работников, которые были представлены в разделе [Представления «Смотритель»](Смотритель#_Представления_).

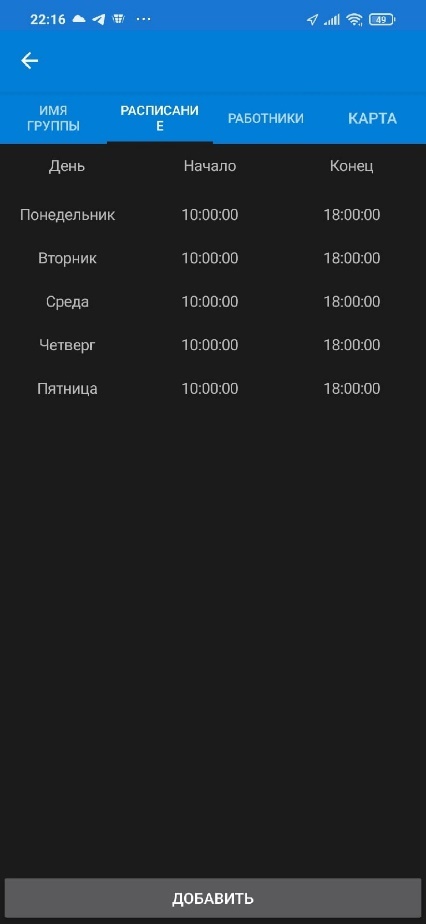
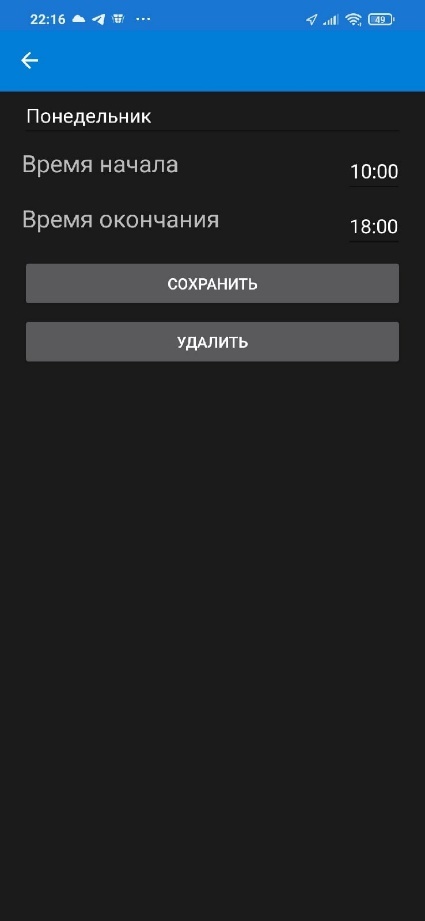
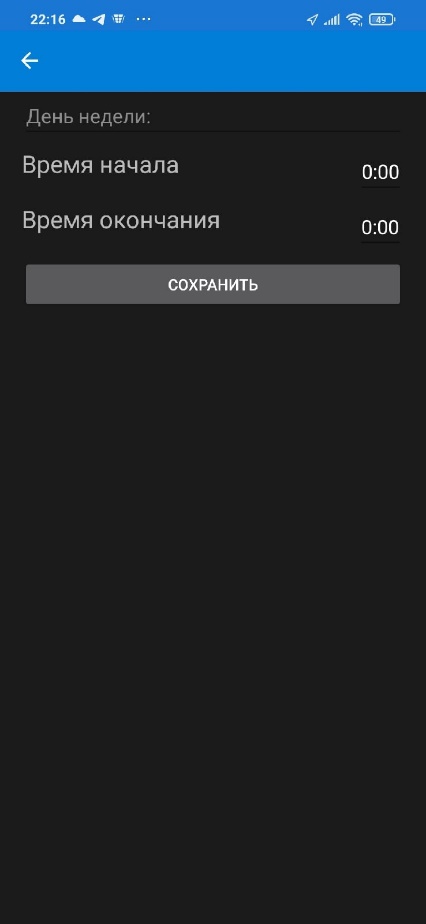
Администратор должен уметь создавать, изменять и удалять рабочие группы. Представления для просмотра группы, ее изменения и удаления представлены на рис. 8. Для того, чтобы перейти от просмотра рабочих групп к редактированию одной из них или удалению, необходимо нажать на выбранную группу. Сразу после этого, приложение переведет пользователя с представления 8a к представлению 8б. В случае, когда администратор захочет создать новую группу, то необходимо на представлении 8а нажать на надпись сверху справа «Добавить». После этого, его переведёт на страницу, указанную на рис 8в. Также при навигации по приложению есть возможность возвращения на более высокий уровень. Для этого необходимо нажать на стрелочку слева-сверху (ее видно на страницах 8б, 8в и прочих).

а) б) в)

Рис. 8 Работа с группами. На рисунке 8а представлен список всех созданных групп. На рисунке 8б представлена форма по изменению и удалению существующей группы. На рисунке 8в представлена страница для создания новой рабочей группы.

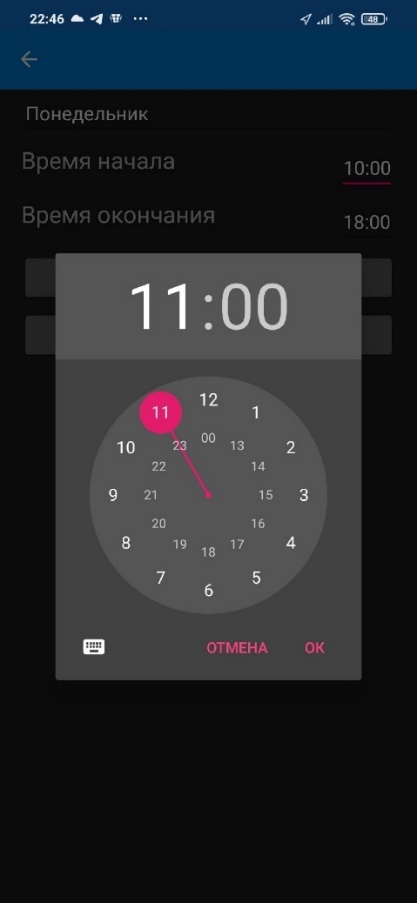
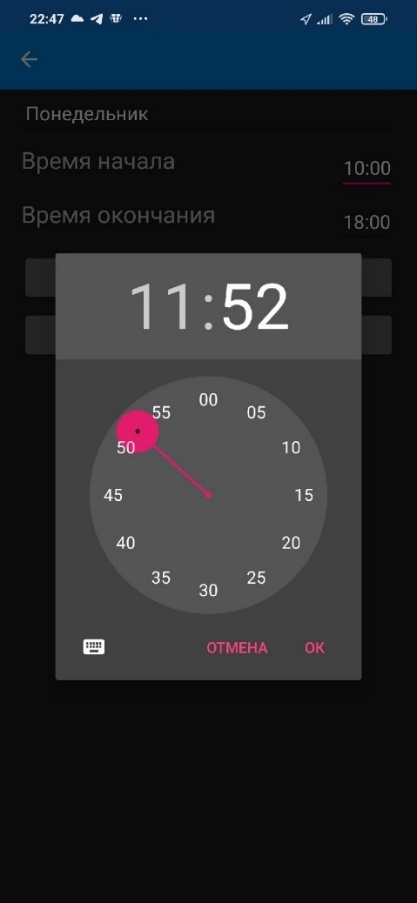
Каждая рабочая группа содержит свое собственное расписание по дням недели с собственным расписанием. Администратор в праве вводить новые рабочие часы на неделе, изменять уже существующее расписание или удалять его. Все представления, которые отвечают за данный функционал представлены на рис. 9. Для добавления нового расписания необходимо нажать на кнопку добавить снизу, после нажатия откроется форма для создания представленная на рис. 9в. В случае необходимости поменять или удалить существующие расписание необходимо нажать на нужную строчку, после чего откроется форма, указанная на рис. 9б.

а) б) в)

Рис. 9 Работа с расписанием рабочей группы. На рисунке 9а представлено расписания для рабочей группы «Разработка и администрирование мобильного приложения». На 9б видна форма для редактирования 1-й записи из списка 9а. На 9в указана форма, которая появляется при нажатии на кнопку «добавить».

Также стоит показать, как выбирать время, на рис. 9б и 9в. Так как там имеет место быть не самый обычное поле текстового ввода. На циферблате необходимо сначала выбрать час, потом нужно выбрать минуту на циферблате. Данный процесс представлен на рис. 10. Также имеется возможность ввести данные через клавиатуру, при нажатии на необходимый элемент управления слева-внизу.

а) б)

Рис. 10 работа с элементом управления для выбора времени. На рис. 10а выбираются часы, на рис. 10б выбираются минуты.

Далее идет раздел для добавления, изменения и удаления работников из рабочей группы. Сам список работников представлен на рис. 11. Формы добавления, изменения и удаления показаны на рис. 12.

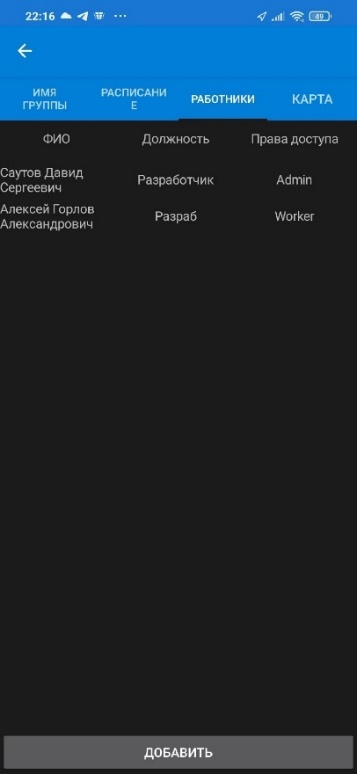
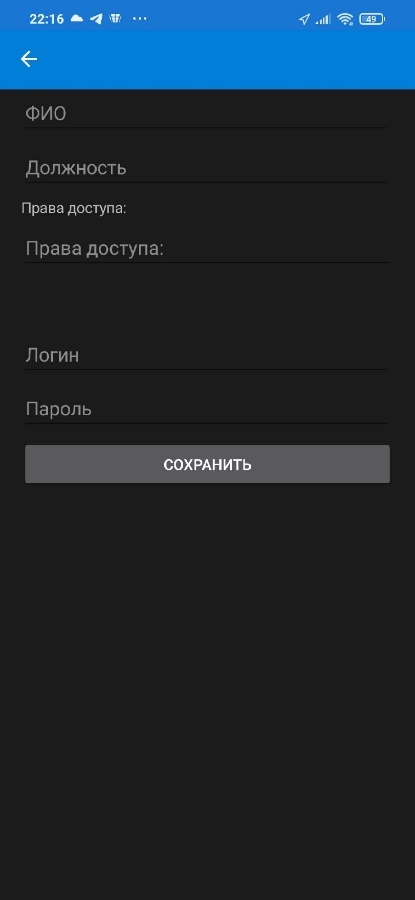
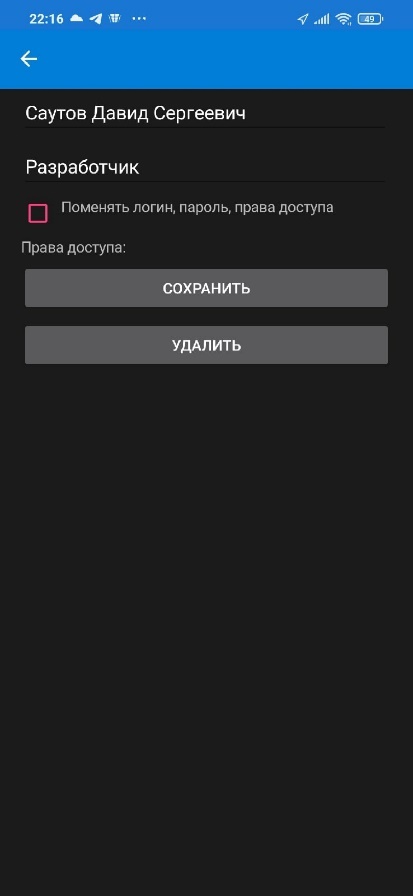
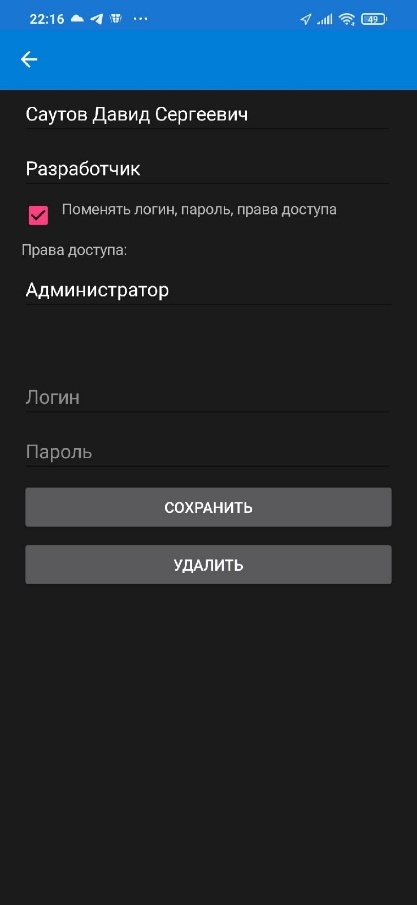


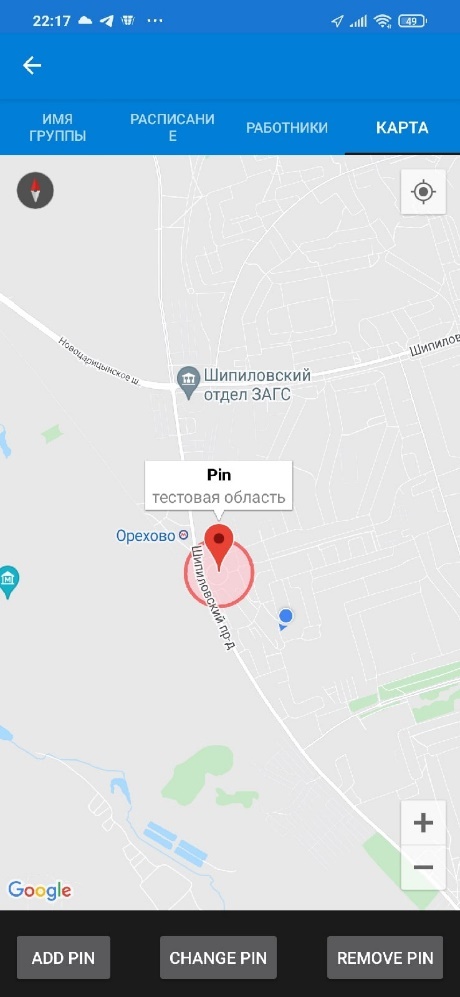
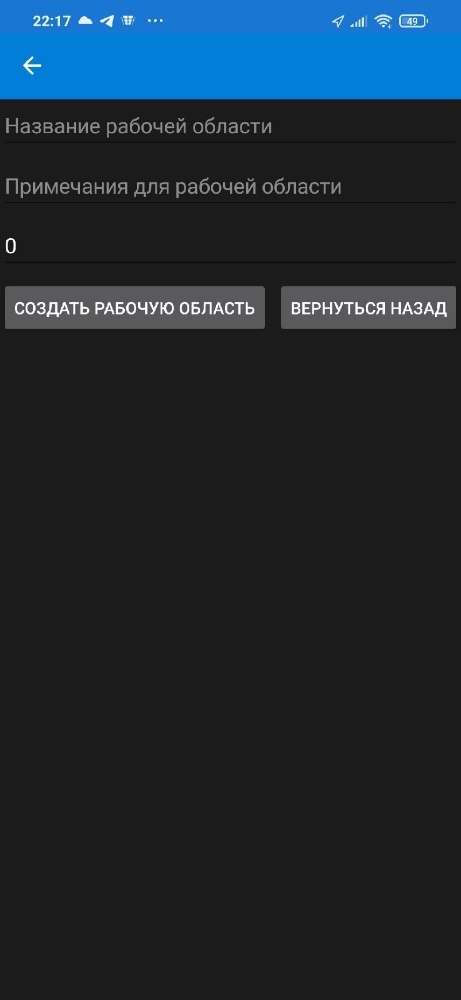
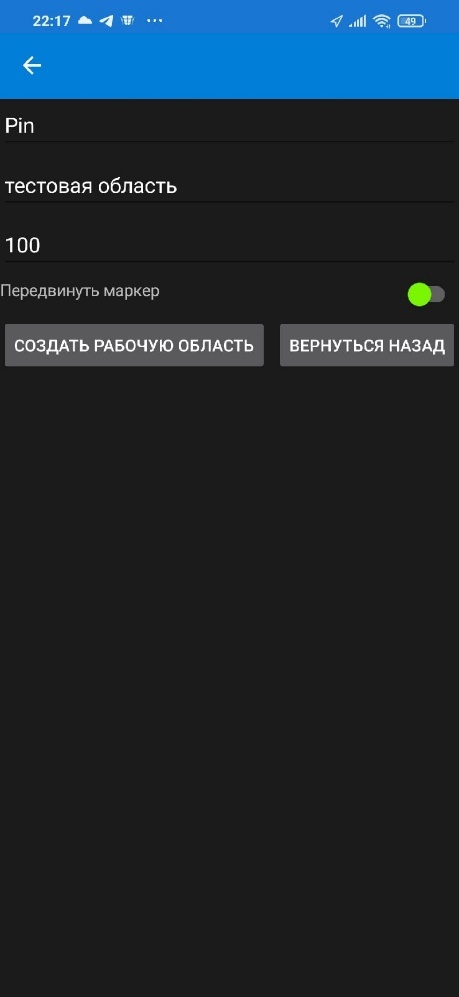
Рис. 11 Список работников в рабочей группе 1.

а) б) в)

Рис. 12. Форма для создания работника представлена на 12а. Форма при изменении данных о пользователе приведена на 12б и 12в.

Последней формой является представление для создания рабочей областей. На рис. 13а представлена карта с обозначенной рабочей областью и информацией о ней, которая появляется при нажатии на маркер. На рис. 13б представлена страница для создания рабочей области, для того чтобы перейти на нее необходимо нажать на соответствующую кнопку «ADD PIN» внизу страницы, после заполнения формы и нажатия на кнопку создания, потом нужно нажать на место на карте, где администратор хочет расположить рабочую область. На рис. 13в форма для редактирования существующего маркера. Для того чтобы перейти к ней, необходимо нажать на кнопку «CHANGE PIN» и нажать на нужный маркер на карте. Для удаления же рабочей области, нужно нажать на соответствующую кнопку, а затем на рабочую область. После чего она удалится.

а) б) в)

Рис. 13. Работа с рабочими областями. На рис. 13а показано как видна рабочая область на карте. На рис. 13б и 13в показаны формы для создания и изменения рабочих областей.

# Реализация и тестирование программы

## Тестирование программы

Тестирование проводилось по ГОСТ 19.301-79

Объект испытаний – мобильное приложение в целом и отдельные методы мобильного приложения.

Цель испытаний – выявление критических и ошибочных ситуаций, а также их обработка.

Средства испытаний – отладчик Visual Studio 2019.

Порядок испытаний – будет реализован следующий порядок испытаний:

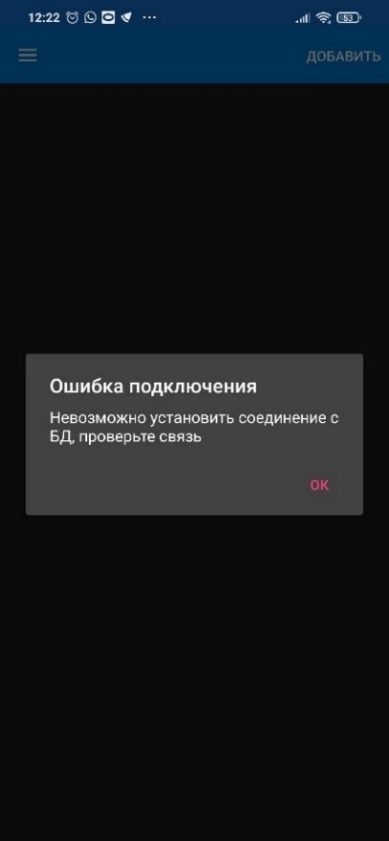
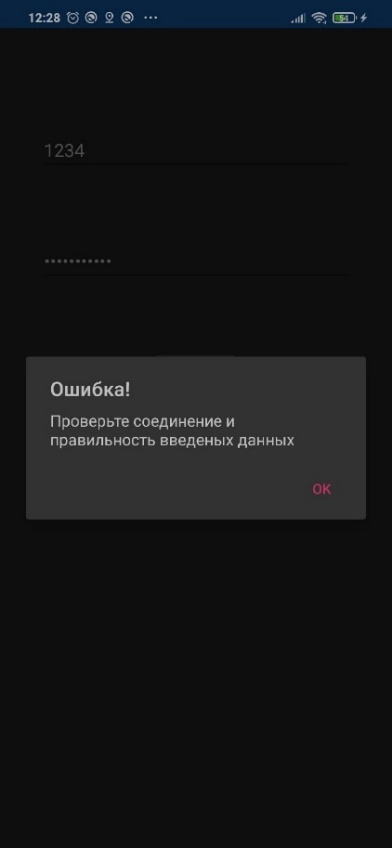
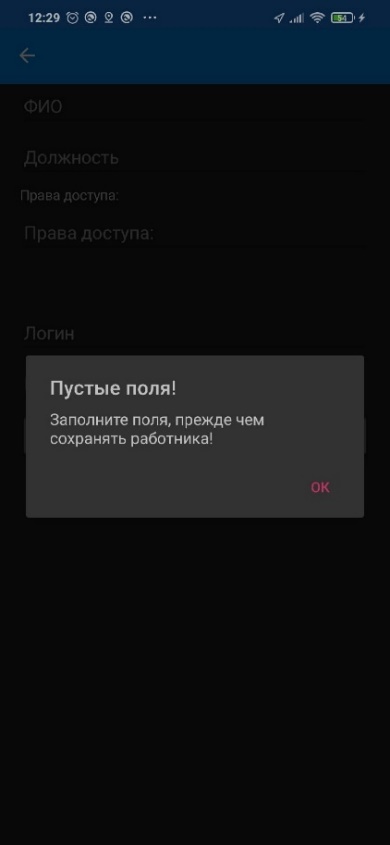
* проверка работоспособности отдельных методов в приложении
* проверка работоспособность приложения в целом

Методы испытаний – метод функционального тестирования.

В ходе тестирования были выявлены следующие исключительные ситуации:

1. Отсутствие соединения с БД не позволяет пользователю получить или отправить данные в БД, из-за чего могут возникать сбои в работе приложения. Решением данной проблемы является проверка соединения с БД во всех функциях, где оно требуется.
2. При вводе неправильных логина и пароля, теперь пользователю высвечивается сообщение об ошибке.
3. При создании или изменении данных о группе и ее составляющих, в случае, когда «Администратор» может ввести не все данные, теперь процесс добавления/изменения не начинает исполняться и высвечивается ошибка о недостаточном объеме данных.

На рис. 14 можно увидеть уведомления, которые видит пользователь при возникновении критической ситуации.

а) б) в)

Рис. 14 Уведомления пользователю при возникновении критических ситуаций. На рис. 14а при отсутствии связи с БД. На рис. 14б при неправильном вводе логина или пароля. На рис. 14в при оставлении пустыми ячеек, при работе «Администратора».

# Заключение

В данной преддипломной практике было разработана БД и мобильное приложение, которое использует эту БД. В дальнейшем планируется реализовать пользовательский интерфейс для «Работника», что будет сделано в ВКР.

# Список литературы

1. **Современные навигационные спутниковые системы**; сайт. – URL: [https://glonass-iac.ru/guide/gnss/](https://glonass-iac.ru/guide/gnss/%20) (дата обращения: 18.05.2022). – Текст: электронный
2. **Xamarin. Документация**; сайт. – URL: <https://docs.microsoft.com/ru-ru/xamarin/get-started/> (дата обращения: 18.05.2022). – Текст: электронный
3. **Xamarin.Forms Map**; сайт. – URL: [https://learntutorials.net/ru/xamarin-forms/topic/3917/](https://learntutorials.net/ru/xamarin-forms/topic/3917/%20) (дата обращения: 18.05.2022). – Текст: электронный
4. Шварц, Б. MySQL по максимуму. Оптимизация, репликация, резервное копировани. 3-е издание /Б. Шварц, П. Зайцев, В. Ткаченко – СПб.: Питер, 2018. – 864 с.: ил. – (Серия «Бестселлеры O’Reilly»).
5. Petzold. Creating Mobile Apps with Xamarin.Forms Preview Edition 2 (Developer Reference) / Charles Petzold.: Microsoft Press, 2015. – 1199 с.: ил.
6. Троелсен, Э. Язык программирования C# 7 и платформы .NET и .NET Core. 8-е издание /Э. Троелсен, Ф. Джепикс – СПб.: ООО «Диалектика», 2018 – 1328 с.: ил. – Парал. Тит. англ.

# Приложение А. Листинг программы.

#### Файл AppShell.xaml

<?xml version="1.0" encoding="utf-8" ?>

<!--

Преддипломная практика

Саутов Давид Сергеевич

А-12-18

Данное представление отвечает за боковое окно навигации

-->

<Shell xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.AppShell"

x:Name="myshell"

xmlns:views="clr-namespace:VKR.Views">

</Shell>

#### Файл AppShell.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class AppShell : Shell

{

public AppShell()

{

InitializeComponent();

Routing.RegisterRoute("Login", typeof(VKR.Views.Login));

MenuItem logoutmenu = new MenuItem()

{

Text = "Logout",

IconImageSource = "icon.png"

};

logoutmenu.Clicked += Logout\_Clikced;

switch (App.DataBase.wtype)

{

case WorkerType.Worker:

{

ShellSection shellSection = new ShellSection();

ShellSection shellSection1 = new ShellSection();

ShellSection shellSection2 = new ShellSection();

shellSection.Title = "Shedule";

shellSection1.Title = "Map";

shellSection2.Title = "About me";

shellSection.Items.Add(new ShellContent()

{

Content = new VKR.Views.Worker.Shedule(),

Route = nameof(Views.Worker.Shedule)

});

shellSection1.Items.Add(new ShellContent()

{

//Заглушка на конструктор карты

Content = new VKR.Views.Worker.Map(),

Route = nameof(Views.Worker.Map)

});

shellSection2.Items.Add(new ShellContent()

{

//Заглушка на конструктор информации о пользователе

Content = new VKR.Views.Worker.MyInformation(),

Route = nameof(Views.Worker.MyInformation)

});

myshell.Items.Add(shellSection);

myshell.Items.Add(shellSection1);

myshell.Items.Add(shellSection2);

myshell.Items.Add(logoutmenu);

myshell.CurrentItem = shellSection;

break;

}

case WorkerType.Watcher:

{

ShellSection shellSection = new ShellSection();

shellSection.Title = "Посещения";

shellSection.Items.Add(new ShellContent()

{

Content = new VKR.Views.Watcher.AllData(),

Route = nameof(Views.Watcher.AllData)

});

myshell.Items.Add(shellSection);

myshell.Items.Add(logoutmenu);

myshell.CurrentItem = shellSection;

break;

}

case WorkerType.Admin:

{

ShellSection shellSection = new ShellSection();

ShellSection shellSection3 = new ShellSection();

shellSection.Title = "Группы";

shellSection3.Title = "Посещения";

shellSection.Items.Add(new ShellContent()

{

Content = new VKR.Views.Admin.Groups(),

Route = nameof(Views.Admin.Groups)

});

shellSection3.Items.Add(new ShellContent()

{

Content = new VKR.Views.Watcher.AllData(),

Route = nameof(Views.Watcher.AllData)

});

myshell.Items.Add(shellSection);

myshell.Items.Add(shellSection3);

myshell.Items.Add(logoutmenu);

myshell.CurrentItem = shellSection;

break;

}

}

}

private async void Logout\_Clikced(object sender, EventArgs e)

{

if (App.DataBase.closeConnection())

{

await Shell.Current.GoToAsync("Login");

Shell.Current.FlyoutIsPresented = false;

}

else

await DisplayAlert("Ошибка!","Вы не можете отключится от БД прямо сейчас!","Ok");

}

}

}

#### Файл AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="com.companyname.vkr">

<uses-sdk android:minSdkVersion="21" android:targetSdkVersion="30" />

<application android:label="VKR.Android" android:theme="@style/MainTheme">

<meta-data android:name="com.google.android.geo.API\_KEY" android:value="AIzaSyDgfoRtm0xpRbhEY1ux5yKR9zsOGbynaHs" />

<meta-data android:name="com.google.android.gms.version" android:value="@integer/google\_play\_services\_version" />

<!-- Necessary for apps that target Android 9.0 or higher -->

<uses-library android:name="org.apache.http.legacy" android:required="false" />

</application>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_BACKGROUND\_LOCATION" />

</manifest>

#### Файл MainActivity.cs

//Данный файл отвечает за работу в приложения на операционных системах Android.

using System;

using Android.App;

using Android.Content.PM;

using Android.Runtime;

using Android.OS;

using Android;

using Android.Content;

namespace VKR.Droid

{

[Activity(Label = "VKR", Icon = "@mipmap/icon", Theme = "@style/MainTheme", MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize )]

public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity

{

const int RequestLocationId = 0;

readonly string[] LocationPermissions =

{

Manifest.Permission.AccessCoarseLocation,

Manifest.Permission.AccessFineLocation,

Manifest.Permission.AccessBackgroundLocation

};

protected override void OnCreate(Bundle savedInstanceState)

{

//this.ShinyOnCreate();

base.OnCreate(savedInstanceState);

Xamarin.Essentials.Platform.Init(this, savedInstanceState);

global::Xamarin.Forms.Forms.Init(this, savedInstanceState);

Xamarin.FormsMaps.Init(this, savedInstanceState);

LoadApplication(new App());

}

protected override void OnStart()

{

base.OnStart();

if ((int)Build.VERSION.SdkInt >= 23)

{

if (CheckSelfPermission(Manifest.Permission.AccessFineLocation) != Permission.Granted)

{

RequestPermissions(LocationPermissions, RequestLocationId);

}

else

{

Console.WriteLine("Location permissions already granted.");

}

}

}

protected override void OnNewIntent(Intent intent)

{

base.OnNewIntent(intent);

//this.ShinyOnNewIntent(intent);

}

protected override void OnActivityResult(int requestCode, Result resultCode, Intent data)

{

base.OnActivityResult(requestCode, resultCode, data);

//this.ShinyOnActivityResult(requestCode, resultCode, data);

}

public override void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)

{

base.OnRequestPermissionsResult(requestCode, permissions, grantResults);

//this.ShinyOnRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == RequestLocationId)

{

if ((grantResults.Length == 1) && (grantResults[0] == (int)Permission.Granted))

{

Console.WriteLine("Location permissions granted.");

}

else

{

Console.WriteLine("Location permissions denied.");

}

}

else

{

Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);

base.OnRequestPermissionsResult(requestCode, permissions, grantResults);

}

}

}

}

#### Файл DataBase.cs

/\*

Данный файл отвечает за соединение с БД, получением, модификацией и удалением данных из БД.

\*/

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using MySqlConnector;

using Newtonsoft.Json;

using VKR.Models;

using VKR.Models.Admin;

using VKR.Models.Watcher;

namespace VKR.Data

{

public class DataBase

{

public WorkerType wtype;

public int emp\_id;

public MySqlConnection connection = new MySqlConnection();

public DataBase() { wtype = WorkerType.NoAuth; }

static string Hash(string input)

{

var hash = new SHA1Managed().ComputeHash(Encoding.UTF8.GetBytes(input));

return string.Concat(hash.Select(b => b.ToString("x2")));

}

public bool standartConnection(string login="common\_user", string password="")

{

if (connection.State == System.Data.ConnectionState.Closed)

try

{

connection = new MySqlConnection($"server=192.168.1.2;port=3306;username={login};password={password};database=vkrdb");

connection.Open();

}

catch

{

return false;

}

return connection.State != System.Data.ConnectionState.Closed;

}

public bool closeConnection()

{

if (connection.State != System.Data.ConnectionState.Closed)

connection.Close();

wtype = WorkerType.NoAuth;

return connection.State == System.Data.ConnectionState.Closed;

}

public bool autorization(string login, string password)

{

if (connection.State == System.Data.ConnectionState.Closed)

if (!standartConnection())

return false;

MySqlCommand command = new MySqlCommand($"SELECT emp\_id, mobile\_status FROM auth\_table WHERE login = '{login}' AND SHA1(SHA1(PASSWORD)) = '{Hash(Hash(password))}'", connection);

MySqlDataReader dr = command.ExecuteReader();

while (dr.Read())

{

emp\_id = (int)dr["emp\_id"];

switch ((string)dr["mobile\_status"])

{

case "Worker":

closeConnection();

standartConnection();

wtype = WorkerType.Worker;

break;

case "Watcher":

closeConnection();

standartConnection("watcher", "mywok/MIvyPWH!)S");

wtype = WorkerType.Watcher;

break;

case "Admin":

closeConnection();

standartConnection("root", "SqlAdmin");

wtype = WorkerType.Admin;

break;

default:

dr.Close();

return false;

}

dr.Close();

return true;

}

dr.Close();

wtype = WorkerType.NoAuth;

return false;

}

public ObservableCollection<WatchWorker> GetWatcher\_Data()

{

ObservableCollection<WatchWorker> lww = new ObservableCollection<WatchWorker>();

string query = "SELECT \* FROM watcher\_all";

if (this.connection.State == System.Data.ConnectionState.Open)

{

MySqlCommand cmd = new MySqlCommand(query, this.connection);

MySqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

lww.Add(new WatchWorker((int)dr["emp\_id"], dr["ФИО"] as string, dr["Должность"] as string, dr["Название подразделения"] as string, (int)dr["Количество пропусков"], (TimeSpan)dr["Время пропусков"]));

}

dr.Close();

foreach (WatchWorker ww in lww)

{

ww.skip\_list = GetWatcher\_Data\_Person(ww.id);

}

}

else

App.Current.MainPage.DisplayAlert("Ошибка подключения", "Невозможно установить соединение с БД, проверьте связь", "Ок");

return lww;

}

public ObservableCollection<Worker\_skip> GetWatcher\_Data\_Person(int id)

{

ObservableCollection<Worker\_skip> skip\_list = new ObservableCollection<Worker\_skip>();

if (connection.State == System.Data.ConnectionState.Open)

{

string query = $"SELECT \* FROM watcher\_table\_skip WHERE emp\_id = {id}";

MySqlCommand cmd = new MySqlCommand(query, App.DataBase.connection);

MySqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

skip\_list.Add(new Worker\_skip((int)dr["id"], ((DateTime)dr["date"]).ToShortDateString(), (TimeSpan)dr["skip\_time"], (TimeSpan)dr["start\_time"], (TimeSpan)dr["end\_time"], (DBNull.Value.Equals(dr["w\_id"])) ? 0 : (int)dr["w\_id"]));

dr.Close();

}

else

App.Current.MainPage.DisplayAlert("Ошибка подключения", "Невозможно установить соединение с БД, проверьте связь", "Ок");

return skip\_list;

}

public bool DeleteWTS(int skip\_id)

{

if (connection.State == System.Data.ConnectionState.Open)

{

string query = $"Delete FROM watcher\_table\_skip WHERE id = {skip\_id}";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

return true;

}

return false;

}

public ObservableCollection<Group> GetAdmin\_Group\_Data()

{

ObservableCollection<Group> Lg = new ObservableCollection<Group>();

string query = "SELECT \* FROM work\_group\_name";

if (this.connection.State == System.Data.ConnectionState.Open)

{

MySqlCommand cmd = new MySqlCommand(query, this.connection);

MySqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

Lg.Add(new Group((int)dr["work\_group\_id"], dr["group\_name"] as string));

}

dr.Close();

}

else

App.Current.MainPage.DisplayAlert("Ошибка подключения", "Невозможно установить соединение с БД, проверьте связь", "Ок");

return Lg;

}

public bool SaveGroup(string groupName, int groupId = -1)

{

if (connection.State == System.Data.ConnectionState.Open)

{

string query = (groupId != - 1) ? $"UPDATE `work\_group\_name` SET `group\_name` = '{groupName}' WHERE `work\_group\_id` = {groupId}" : $"INSERT INTO `work\_group\_name` (`work\_group\_id`, `group\_name`) VALUES (NULL, '{groupName}')";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

return true;

}

return false;

}

public bool DeleteGroup(int groupId)

{

if (connection.State == System.Data.ConnectionState.Open)

{

string query = $"Delete FROM work\_group\_name WHERE work\_group\_id = {groupId}";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

return true;

}

return false;

}

public ObservableCollection<Shedule> GetShedules(int groupId)

{

ObservableCollection<Shedule> ls = new ObservableCollection<Shedule>();

string query = $"SELECT \* FROM time\_table WHERE work\_group\_id = {groupId}";

if (this.connection.State == System.Data.ConnectionState.Open)

{

MySqlCommand cmd = new MySqlCommand(query, this.connection);

MySqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

ls.Add(new Shedule((int)dr["id"], (int)dr["week\_day"], (TimeSpan)dr["time\_start"], (TimeSpan)dr["time\_end"]));

}

dr.Close();

}

else

App.Current.MainPage.DisplayAlert("Ошибка подключения", "Невозможно установить соединение с БД, проверьте связь", "Ок");

return ls;

}

public bool SaveShedule(Shedule shedule, int groupId)

{

if (connection.State == System.Data.ConnectionState.Open)

{

string query = (shedule.id != -1) ? $"UPDATE `time\_table` SET `week\_day` = '{(int)shedule.weekDay}', `time\_start` = '{shedule.time\_start\_str}', `time\_end` = '{shedule.time\_end\_str}' WHERE `id` = {shedule.id}"

: $"INSERT INTO `time\_table` VALUES (NULL, '{groupId}', '{(int)shedule.weekDay}','{shedule.time\_start\_str}','{shedule.time\_end\_str}')";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

query = $"DROP EVENT IF EXISTS `wg\_{groupId}\_{(int)shedule.weekDay}`; CREATE DEFINER =`root`@`%` EVENT `wg\_{groupId}\_{(int)shedule.weekDay}` ON SCHEDULE EVERY 1 WEEK STARTS '2022-04-0{3+((int)shedule.weekDay)} {shedule.time\_end\_str}' ON COMPLETION PRESERVE ENABLE DO CALL proc\_skipt0(1, '{shedule.time\_start\_str}', '{shedule.time\_end\_str}')";

cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

return true;

}

return false;

}

public bool DeleteShedule(Shedule shedule, int groupId)

{

if (connection.State == System.Data.ConnectionState.Open)

{

string query = $"Delete FROM time\_table WHERE id = {shedule.id}";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

query = $"DROP EVENT IF EXISTS `wg\_{groupId}\_{(int)shedule.weekDay}`;";

cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

return true;

}

return false;

}

public ObservableCollection<Employee> GetAdminWorkers(int groupId)

{

ObservableCollection<Employee> le = new ObservableCollection<Employee>();

string query = $"SELECT \* FROM admin\_employeers WHERE work\_group\_id = {groupId}";

if (this.connection.State == System.Data.ConnectionState.Open)

{

MySqlCommand cmd = new MySqlCommand(query, this.connection);

MySqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

le.Add(new Employee((int)dr["emp\_id"], (string)dr["emp\_fio"], (string)dr["emp\_pos"], (string)dr["mobile\_status"]));

}

dr.Close();

}

else

App.Current.MainPage.DisplayAlert("Ошибка подключения", "Невозможно установить соединение с БД, проверьте связь", "Ок");

return le;

}

public bool SaveEmployee(Employee employee, int groupId)

{

if (connection.State == System.Data.ConnectionState.Open)

{

if (employee.id == -1)

{

string query = $"INSERT INTO `employee` VALUES (NULL, '{employee.FIO}', '{employee.Position}','{groupId}');";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

query = $"SELECT `emp\_id` FROM `employee` WHERE emp\_fio = '{employee.FIO}' AND emp\_pos = '{employee.Position}' AND `work\_group\_id` = '{groupId}';";

cmd = new MySqlCommand(query, connection);

employee.id = (int)cmd.ExecuteScalar();

query = $"INSERT INTO `auth\_table` VALUES ('{employee.id}', '{employee.Login}', '{employee.Password}', '{employee.wt\_str}');";

cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

}

else

{

string query = $"UPDATE `employee` SET `emp\_fio` = '{employee.FIO}', `emp\_pos` = '{employee.Position}' WHERE `emp\_id` = {employee.id};";

if (!(string.IsNullOrEmpty(employee.Login) || string.IsNullOrEmpty(employee.Password)))

{

query += $" UPDATE `auth\_table` SET `login` = '{employee.Login}', `password` = '{employee.Password}, `mobile\_status` = '{employee.wt\_str}' WHERE `emp\_id` = {employee.id}';";

}

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

}

return true;

}

return false;

}

public bool DeleteEmployee(Employee employee)

{

if (connection.State == System.Data.ConnectionState.Open)

{

string query = $"Delete FROM employee WHERE `emp\_id` = '{employee.id}'";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

return true;

}

return false;

}

public ObservableCollection<Marker> GetMarkers(int groupId)

{

ObservableCollection<Marker> lm = new ObservableCollection<Marker>();

string query = $"SELECT \* FROM marker\_table WHERE group\_id = {groupId}";

if (this.connection.State == System.Data.ConnectionState.Open)

{

MySqlCommand cmd = new MySqlCommand(query, this.connection);

MySqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

Marker buff = JsonConvert.DeserializeObject<Marker>((string)dr["Marker"]);

buff.uid = (int)dr["marker\_id"];

lm.Add(buff);

}

dr.Close();

}

else

App.Current.MainPage.DisplayAlert("Ошибка подключения", "Невозможно установить соединение с БД, проверьте связь", "Ок");

return lm;

}

public bool SaveMarker(Marker marker, int groupId)

{

if (connection.State == System.Data.ConnectionState.Open)

{

if (marker.uid == -1)

{

string query = $"INSERT INTO `marker\_table` VALUES (NULL, '{groupId}', '{JsonConvert.SerializeObject(marker)}');";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

}

else

{

string query = $"UPDATE `marker\_table` SET `Marker` = '{JsonConvert.SerializeObject(marker)}' WHERE `marker\_id` = {marker.uid};";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

}

return true;

}

return false;

}

public bool DeleteMarker(Marker marker)

{

if (connection.State == System.Data.ConnectionState.Open)

{

string query = $"Delete FROM `marker\_table` WHERE `marker\_id` = '{marker.uid}'";

MySqlCommand cmd = new MySqlCommand(query, connection);

cmd.ExecuteNonQuery();

return true;

}

return false;

}

}

}

#### Файл Group.cs

//Данный класс является моделью Group для представлений Администратора

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Text;

namespace VKR.Models.Admin

{

public enum WeekDay

{

Понедельник = 1,

Вторник = 2,

Среда = 3,

Четверг = 4,

Пятница = 5,

Суббота = 6,

Воскресенье = 0

}

public class Shedule

{

public int id;

public WeekDay weekDay { get; set; }

public TimeSpan time\_start { get; set; }

public string time\_start\_str { get { return time\_start.ToString(); }}

public TimeSpan time\_end { get; set; }

public string time\_end\_str { get { return time\_end.ToString(); } }

public Shedule(int id, int weekDay, TimeSpan time\_start, TimeSpan time\_end)

{

this.id = id;

this.weekDay = (WeekDay)weekDay;

this.time\_start = time\_start;

this.time\_end = time\_end;

}

public Shedule()

{

id = -1;

}

}

public class Employee

{

public int id;

public string FIO { get; set; }

public string Position { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public WorkerType wt { get; set; }

public string wt\_str { get { return wt.ToString(); } }

public Employee(int id, string FIO, string Position, string wt)

{

this.id = id;

this.FIO = FIO;

this.Position = Position;

this.wt = (WorkerType)Enum.Parse(typeof(WorkerType), wt);

}

public Employee()

{

id = -1;

}

}

public class Group : INotifyPropertyChanged

{

public int GroupId { get; set; }

public string NameOfGroup { get; set; }

public ObservableCollection<Shedule> LS { get { return \_LS; } set { \_LS = value; OnPropertyChanged("LS"); } }

private ObservableCollection<Shedule> \_LS { get; set; }

public ObservableCollection<Employee> LE { get { return \_LE; } set { \_LE = value; OnPropertyChanged("LE"); } }

private ObservableCollection<Employee> \_LE { get; set; }

public ObservableCollection<Marker> LM { get { return \_LM; } set { \_LM = value; OnPropertyChanged("LM"); } }

private ObservableCollection<Marker> \_LM { get; set; }

public Group()

{

this.GroupId = -1;

}

public Group(int NumberOfGroup, string NameOfGroup)

{

this.GroupId = NumberOfGroup;

this.NameOfGroup = NameOfGroup;

//Заглушка на расписание

//Заглушка на пользователей

//Заглушка на маркеры

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

}

}

#### Файл WatchWorker.cs

using System;

using System.Collections.Generic;

using System.Text;

using MySqlConnector;

using System.Collections.ObjectModel;

using Newtonsoft.Json;

using System.ComponentModel;

namespace VKR.Models.Watcher

{

public class Worker\_skip

{

public int marker\_id { get; set; }

public int skip\_id { get; set; }

public string date { get; set; }

public string skip\_time { get; set; }

public string start\_time { get; set; }

public string end\_time { get; set; }

[JsonConstructor]

public Worker\_skip(int skip\_id, string date, TimeSpan skip\_time, TimeSpan start\_time, TimeSpan end\_time, int marker\_id = 0)

{

this.skip\_id = skip\_id;

this.date = date;

this.skip\_time = skip\_time.ToString();

this.start\_time = start\_time.ToString();

this.end\_time = end\_time.ToString();

this.marker\_id = marker\_id;

}

}

public class WatchWorker : INotifyPropertyChanged

{

[JsonConstructor]

public WatchWorker(int id, string fio, string position, string workGroup, int numberOfAbsence, TimeSpan sumTime)

{

this.id = id;

this.FIO = fio;

this.Position = position;

this.WorkGroup = workGroup;

this.NumberOfAbsence = numberOfAbsence;

this.SumTimeOfAbsence = sumTime.ToString();

//skip\_list = App.DataBase.GetWatcher\_Data\_Person(this.id);

dlt\_rows = Xamarin.Forms.SelectionMode.Single;

}

public WatchWorker(int id)

{

string query = $"SELECT \* FROM watcher\_all WHERE emp\_id = {id}";

if (App.DataBase.connection.State == System.Data.ConnectionState.Open)

{

MySqlCommand cmd = new MySqlCommand(query, App.DataBase.connection);

MySqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

FIO = dr["ФИО"] as string;

Position = dr["Должность"] as string;

WorkGroup = dr["Название подразделения"] as string;

NumberOfAbsence = (int)dr["Количество пропусков"];

SumTimeOfAbsence = ((TimeSpan)dr["Время пропусков"]).ToString();

}

dr.Close();

}

skip\_list = App.DataBase.GetWatcher\_Data\_Person(this.id);

dlt\_rows = Xamarin.Forms.SelectionMode.None;

}

public ObservableCollection<Worker\_skip> skip\_list { get { return \_skip\_list; } set { \_skip\_list = value; OnPropertyChanged("skip\_list"); } }

public Xamarin.Forms.SelectionMode dlt\_rows { get; private set; }

public int id { get { return \_id; } set { \_id = value; OnPropertyChanged("id"); } }

public string FIO { get { return \_FIO; } set { \_FIO = value; OnPropertyChanged("FIO"); } }

public string Position { get { return \_Position; } set { \_Position = value; OnPropertyChanged("Position"); } }

public string WorkGroup { get { return \_WorkGroup; } set { \_WorkGroup = value; OnPropertyChanged("WorkGroup"); } }

public int NumberOfAbsence { get { return \_NumberOfAbsence; } set { \_NumberOfAbsence = value; OnPropertyChanged("NumberOfAbsence"); } }

public string SumTimeOfAbsence { get { return \_SumTimeOfAbsence; } set { \_SumTimeOfAbsence = value; OnPropertyChanged("SumTimeOfAbsence"); } }

private ObservableCollection<Worker\_skip> \_skip\_list { get; set; }

private Xamarin.Forms.SelectionMode \_dlt\_rows { get; set; }

private int \_id { get; set; }

private string \_FIO { get; set; }

private string \_Position { get; set; }

private string \_WorkGroup { get; set; }

private int \_NumberOfAbsence { get; set; }

private string \_SumTimeOfAbsence { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

}

}

#### Файл Marker.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Text;

using Xamarin.Forms.Maps;

using Xamarin.Forms;

using Position = Xamarin.Forms.Maps.Position;

using Distance = Xamarin.Forms.Maps.Distance;

namespace VKR.Models

{

public class Marker

{

public int uid;

public Pin pin;

public Circle circle;

//public GeofenceRegion region;

public Marker()

{

uid = -1;

pin = new Pin();

circle = new Circle();

circle.StrokeColor = Color.FromHex("#88FF0000");

circle.StrokeWidth = 8;

circle.FillColor = Color.FromHex("#88FFC0CB");

pin.Type = PinType.Place;

}

public Marker(Distance radius, string title, string text)

{

uid = -1;

pin = new Pin();

circle = new Circle();

circle.Radius = radius;

circle.StrokeColor = Color.FromHex("#88FF0000");

circle.StrokeWidth = 8;

circle.FillColor = Color.FromHex("#88FFC0CB");

pin.Label = title;

pin.Address = text;

pin.Type = PinType.Place;

}

public void setPosition(Position position)

{

pin.Position = position;

circle.Center = position;

}

/\* public void setRegion(GeofenceRegion region)

{

this.region = region

}\*/

[JsonConstructor]

public Marker(int uid, Pin pin, Circle circle/\*, GeofenceRegion region\*/)

{

this.uid = uid;

this.pin = pin;

this.circle = circle;

circle.StrokeColor = Color.FromHex("#88FF0000");

circle.StrokeWidth = 8;

circle.FillColor = Color.FromHex("#88FFC0CB");

pin.Type = PinType.Place;

//this.region = region;

}

}

}

#### Файл MapPageViewModel

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Text;

using System.Windows.Input;

using VKR.Models;

using VKR.Views.Admin;

using Xamarin.Forms;

using Xamarin.Forms.Maps;

namespace VKR.ViewModels.Admin

{

public enum ButtonClickedType

{

Add,

Change,

Delete,

None

}

class MapPageViewModel

{

public MapPageViewModel()

{

m = null;

flag = false;

buttonClickedType = ButtonClickedType.None;

map.MoveToLastRegionOnLayoutChange

= map.IsShowingUser

= map.HasZoomEnabled

= map.HasScrollEnabled

= true;

map.MapClicked += Map\_MapClicked;

map.PropertyChanged += Map\_PropertyChanged;

if (Application.Current.Properties.ContainsKey("currentMapPosition"))

map.MoveToRegion(JsonConvert.DeserializeObject<MapSpan>((string)Application.Current.Properties["currentMapPosition"]));

mlist = App.DataBase.GetMarkers(groupId);

foreach (Marker marker in mlist)

{

marker.pin.MarkerClicked += Pin\_MarkerClicked;

map.Pins.Add(marker.pin);

map.MapElements.Add(marker.circle);

//заглушка на shiny

}

this.AddPin = new Command(async () =>

{

m = null;

buttonClickedType = ButtonClickedType.Add;

await Application.Current.MainPage.Navigation.PushAsync(new MarkerCreate());

});

this.ChangePin = new Command(() => { buttonClickedType = ButtonClickedType.Change; });

this.DeletePin = new Command(() => { buttonClickedType = ButtonClickedType.Delete; });

}

public async void Pin\_MarkerClicked(object sender, PinClickedEventArgs e)

{

Pin pin = (Pin)sender;

switch (buttonClickedType)

{

case ButtonClickedType.Change:

{

foreach (Marker marker in mlist)

{

if (marker.pin == pin)

{

m = marker;

break;

}

}

await Application.Current.MainPage.Navigation.PushAsync(new MarkerCreate());

}

break;

case ButtonClickedType.Delete:

{

foreach (Marker marker in mlist)

{

if (marker.pin == pin)

{

m = marker;

break;

}

}

App.DataBase.DeleteMarker(m);

map.Pins.Remove(MapPageViewModel.m.pin);

map.MapElements.Remove(MapPageViewModel.m.circle);

mlist.Remove(m);

buttonClickedType = ButtonClickedType.None;

m = null;

}

break;

}

}

private void Map\_PropertyChanged(object sender, PropertyChangedEventArgs e)

{

if (map.VisibleRegion != null)

Application.Current.Properties["currentMapPosition"] = JsonConvert.SerializeObject(new MapSpan(map.VisibleRegion.Center, map.VisibleRegion.LatitudeDegrees, map.VisibleRegion.LongitudeDegrees));

}

private async void Map\_MapClicked(object sender, MapClickedEventArgs e)

{

if (flag)

{

m.setPosition(e.Position);

m.pin.MarkerClicked += Pin\_MarkerClicked;

map.Pins.Add(m.pin);

map.MapElements.Add(m.circle);

App.DataBase.SaveMarker(m, groupId);

mlist = App.DataBase.GetMarkers(groupId);

//заглушка

//создание региона

m = null;

buttonClickedType = ButtonClickedType.None;

flag = false;

}

}

public ICommand AddPin { get; }

public ICommand ChangePin { get; }

public ICommand DeletePin { get; }

public static Marker m;

public static int groupId { get; set; }

public static Map map { get; set; }

public ObservableCollection<Marker> mlist { get; set; }

public static ButtonClickedType buttonClickedType { get; set; }

public static bool flag { get; set; }

}

}

#### Файл MarkerCreateViewModel

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Windows.Input;

using VKR.Models;

using Xamarin.Forms;

namespace VKR.ViewModels.Admin

{

public class MarkerCreateViewModel : INotifyPropertyChanged

{

public Marker marker;

public MarkerCreateViewModel()

{

marker = (MapPageViewModel.m == null) ? new Marker() : MapPageViewModel.m;

this.CreateGeofence = new Command(async () =>

{

if (string.IsNullOrEmpty(MTitle))

{

await App.Current.MainPage.DisplayAlert("Нет данных!", "Необходимо заполнит название рабочей области!", "Ок!");

return;

}

if (string.IsNullOrEmpty(MText))

{

await App.Current.MainPage.DisplayAlert("Нет данных!", "Необходимо заполнит примечания для рабочей области!", "Ок!");

return;

}

if (MRadius<100)

{

await App.Current.MainPage.DisplayAlert("Некорректные данные!", "Радиус не должен быть меньше 100", "Ок!");

return;

}

if (MapPageViewModel.buttonClickedType == ButtonClickedType.Change)

{

MapPageViewModel.map.Pins.Remove(MapPageViewModel.m.pin);

MapPageViewModel.map.MapElements.Remove(MapPageViewModel.m.circle);

//Удаление геозоны

if (!movemarker)

{

MapPageViewModel.map.Pins.Add(marker.pin);

MapPageViewModel.map.MapElements.Add(marker.circle);

App.DataBase.SaveMarker(marker, MapPageViewModel.groupId);

MapPageViewModel.buttonClickedType = ButtonClickedType.None;

//создание геозоны

await App.Current.MainPage.Navigation.PopAsync();

return;

}

}

MapPageViewModel.m = marker;

MapPageViewModel.flag = true;

await App.Current.MainPage.Navigation.PopAsync();

}

);

this.GoBack = new Command(async () =>

{

MapPageViewModel.buttonClickedType = ButtonClickedType.None;

MapPageViewModel.m = null;

await App.Current.MainPage.Navigation.PopAsync();

});

}

public ICommand CreateGeofence { get; }

public ICommand GoBack { get; }

public string MTitle

{

get { return marker.pin.Label; }

set { marker.pin.Label = value; OnPropertyChanged("MTitle"); }

}

public string MText

{

get { return marker.pin.Address; }

set { marker.pin.Address = value; OnPropertyChanged("MText"); }

}

public int MRadius

{

get { return (int)marker.circle.Radius.Meters; }

set { marker.circle.Radius = new Xamarin.Forms.Maps.Distance(value); OnPropertyChanged("MRadius"); }

}

bool movemarker;

public bool MoveMarker

{

get { return movemarker; }

set { movemarker = value; OnPropertyChanged("MoveMarker"); }

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

}

}

#### Файл LoginViewModel

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel;

using VKR.ViewModels;

using System.Windows.Input;

using Xamarin.Forms;

namespace VKR.ViewModels

{

public class LoginViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

string login;

public string Login

{

get { return login; }

set

{

if (login != value)

{

login = value;

OnPropertyChanged("Login");

}

}

}

string password;

public string Password

{

get { return password; }

set

{

if (password != value)

{

password = value;

OnPropertyChanged("Password");

}

}

}

public ICommand LoginCommand { protected set; get; }

public LoginViewModel()

{

LoginCommand = new Command(auth);

}

private async void auth()

{

App.DataBase.closeConnection();

if (this.Login != null && this.Password != null)

if (App.DataBase.autorization(this.Login, this.Password))

App.Current.MainPage = new AppShell();

else

await Application.Current.MainPage.DisplayAlert("Ошибка!", "Проверьте соединение и правильность введеных данных", "Ok");

else

await Application.Current.MainPage.DisplayAlert("Ошибка!", "Не все поля заполнены!", "Ok");

}

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

}

}

#### Файл EmployeerCreate.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Admin.EmployeerCreate">

<ContentPage.Content>

<StackLayout Margin="20,0">

<Entry Text="{Binding FIO}" Placeholder="ФИО" HorizontalOptions="StartAndExpand" WidthRequest="350"/>

<Entry Text="{Binding Position}" Placeholder="Должность" HorizontalOptions="StartAndExpand" WidthRequest="350"/>

<StackLayout Orientation="Horizontal" x:Name="check">

<CheckBox CheckedChanged="CheckBox\_CheckedChanged" x:Name="checkbox"/>

<Label Text="Поменять логин, пароль, права доступа"/>

</StackLayout>

<Label Text="Права доступа: "/>

<Picker Title="Права доступа:" x:Name="picker" Margin="0,0,0,50">

<Picker.Items>

<x:String>Работник</x:String>

<x:String>Смотритель</x:String>

<x:String>Администратор</x:String>

</Picker.Items>

</Picker>

<Entry Text="{Binding Login}" Placeholder="Логин" HorizontalOptions="StartAndExpand" WidthRequest="350" x:Name="Login"/>

<Entry Text="{Binding Password}" Placeholder="Пароль" HorizontalOptions="StartAndExpand" WidthRequest="350" x:Name="Password" IsPassword="True"/>

<Button Text="Сохранить" Clicked="SaveButtonClicked"/>

<Button Text="Удалить" Clicked="DeleteButtonClicked" x:Name="deleteb"/>

</StackLayout>

</ContentPage.Content>

</ContentPage>

#### Файл EmployeerCreate.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR.Views.Admin

{

public partial class EmployeerCreate : ContentPage

{

VKR.Models.Admin.Employee emp;

int GroupId;

public EmployeerCreate(ref VKR.Models.Admin.Employee emp, int GroupId)

{

this.emp = emp;

this.GroupId = GroupId;

InitializeComponent();

if (emp.id == -1)

{

deleteb.IsEnabled = deleteb.IsVisible = check.IsVisible = check.IsEnabled = false;

checkbox.IsChecked = picker.IsEnabled = picker.IsVisible = Login.IsEnabled = Login.IsVisible = Password.IsEnabled = Password.IsVisible = true;

}

else

{

switch (emp.wt)

{

case WorkerType.Worker:

picker.SelectedIndex = 0;

break;

case WorkerType.Watcher:

picker.SelectedIndex = 1;

break;

case WorkerType.Admin:

picker.SelectedIndex = 2;

break;

default:

picker.SelectedIndex = -1;

break;

}

deleteb.IsEnabled = deleteb.IsVisible = check.IsVisible = check.IsEnabled = true;

picker.IsEnabled = picker.IsVisible = Login.IsEnabled = Login.IsVisible = Password.IsEnabled = Password.IsVisible = false;

}

}

protected override void OnAppearing()

{

base.OnAppearing();

this.BindingContext = emp;

}

private async void SaveButtonClicked(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(emp.FIO) && !string.IsNullOrEmpty(emp.Position) && (!checkbox.IsChecked || (!string.IsNullOrEmpty(emp.Login) && !string.IsNullOrEmpty(emp.Password) && picker.SelectedIndex != -1)))

{

switch (picker.SelectedIndex)

{

case 0:

emp.wt = WorkerType.Worker;

break;

case 1:

emp.wt = WorkerType.Watcher;

break;

case 2:

emp.wt = WorkerType.Admin;

break;

default:

break;

}

App.DataBase.SaveEmployee(emp, GroupId);

await Navigation.PopAsync();

}

else

await DisplayAlert("Пустые поля!", "Заполните поля, прежде чем сохранять работника!", "Ок");

}

private async void DeleteButtonClicked(object sender, EventArgs e)

{

var ans = await DisplayAlert("Удаление пользователя", "Вы точно хотите удалить пользователя?", "Да", "Нет");

if (ans == true)

{

App.DataBase.DeleteEmployee(emp);

await Navigation.PopAsync();

}

}

private void CheckBox\_CheckedChanged(object sender, CheckedChangedEventArgs e)

{

picker.IsEnabled = picker.IsVisible = Login.IsEnabled = Login.IsVisible = Password.IsEnabled = Password.IsVisible = ((CheckBox)sender).IsChecked;

}

}

}

#### Файл Employeers.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Admin.Employeers"

Title="Работники">

<StackLayout Orientation="Vertical" Spacing="5">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="125" />

<ColumnDefinition Width="125" />

<ColumnDefinition Width="125" />

</Grid.ColumnDefinitions>

<Label Text="ФИО"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="1"

Text="Должность"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="2"

Text="Права доступа"

HorizontalOptions="Center"

VerticalOptions="Center" />

</Grid>

<CollectionView SelectionMode="Single"

SelectionChanged="CollectionView\_SelectionChanged"

x:Name="collView">

<CollectionView.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="125" />

<ColumnDefinition Width="125" />

<ColumnDefinition Width="125" />

</Grid.ColumnDefinitions>

<Label Text="{Binding FIO}"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="1"

Text="{Binding Position}"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="2"

Text="{Binding wt\_str}"

HorizontalOptions="Center"

VerticalOptions="Center" />

</Grid>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

<Button Text="Добавить" Clicked="AddToolbar\_Clicked" VerticalOptions="End"/>

</StackLayout>

</ContentPage>

#### Файл Employeers.xaml.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR.Views.Admin

{

public partial class Employeers : ContentPage, INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

private VKR.Models.Admin.Group \_group

{

get; set;

}

public VKR.Models.Admin.Group group

{

get

{

return \_group;

}

set

{

\_group = value;

OnPropertyChanged("LE");

OnPropertyChanged("group");

}

}

public Employeers(ref VKR.Models.Admin.Group group)

{

group.LE = App.DataBase.GetAdminWorkers(group.GroupId);

this.group = group;

InitializeComponent();

collView.SetBinding(ItemsView.ItemsSourceProperty, nameof(this.group.LE));

}

protected override void OnAppearing()

{

base.OnAppearing();

group.LE = App.DataBase.GetAdminWorkers(group.GroupId);

this.BindingContext = group;

}

private async void AddToolbar\_Clicked(object sender, EventArgs e)

{

VKR.Models.Admin.Employee s = new VKR.Models.Admin.Employee();

await Navigation.PushAsync(new EmployeerCreate(ref s, group.GroupId));

}

private async void CollectionView\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (e.CurrentSelection != null)

{

VKR.Models.Admin.Employee employee = e.CurrentSelection.FirstOrDefault() as VKR.Models.Admin.Employee;

await Navigation.PushAsync(new EmployeerCreate(ref employee, group.GroupId));

}

}

}

}

#### Файл Group.xaml

<?xml version="1.0" encoding="utf-8" ?>

<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Admin.Group"

x:Name="tabbedpage">

</TabbedPage>

#### Файл Group.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR.Views.Admin

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class Group : TabbedPage

{

VKR.Models.Admin.Group group;

public Group(ref VKR.Models.Admin.Group group)

{

this.group = group;

InitializeComponent();

Children.Add(new GroupCreate(ref this.group));

Children.Add(new TimeShedule(ref this.group));

Children.Add(new Employeers(ref this.group));

Children.Add(new MapPage(ref this.group));

//Children.Add();

}

}

}

#### Файл GroupCreate.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Admin.GroupCreate"

Title="Имя группы">

<ContentPage.Content>

<StackLayout>

<StackLayout Margin="20">

<Editor Placeholder="Наименование рабочей группы" Text="{Binding NameOfGroup}" HeightRequest="80"/>

<Grid ColumnDefinitions="\*,\*">

<Button Text="Сохранить" Clicked="OnSaveButtonClicked"/>

<Button Grid.Column="1" Text="Удалить" Clicked="OnDeleteButtonClicked" x:Name="dButton"/>

</Grid>

</StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

#### Файл GroupCreate.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR.Views.Admin

{

public partial class GroupCreate : ContentPage

{

VKR.Models.Admin.Group group;

public GroupCreate(ref VKR.Models.Admin.Group group)

{

InitializeComponent();

this.BindingContext = group;

this.group = group;

if (group.GroupId == -1)

{

dButton.IsVisible = dButton.IsEnabled = false;

}

}

private async void OnSaveButtonClicked(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(group.NameOfGroup))

{

App.DataBase.SaveGroup(group.NameOfGroup, group.GroupId);

await Navigation.PopAsync();

}

else

await DisplayAlert("Пустые поля!", "Для добавление в базу данных новой группы необходимо заполнить все поля!", "Ок");

}

private async void OnDeleteButtonClicked(object sender, EventArgs e)

{

var ans = await DisplayAlert("Удаление группы", "Удалить группу? Все связанные с ней данные так же будут удалены!", "Да", "Нет");

if (ans == true)

{

App.DataBase.DeleteGroup(group.GroupId);

await Navigation.PopAsync();

}

}

}

}

#### Файл Groups.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Admin.Groups">

<ContentPage.ToolbarItems>

<ToolbarItem Text="Добавить" Clicked="AddToolbar\_Clicked"/>

</ContentPage.ToolbarItems>

<CollectionView ItemsSource="{Binding LG}"

SelectionMode="Single"

SelectionChanged="collview\_SelectionChanged"

x:Name="collview">

<CollectionView.ItemTemplate>

<DataTemplate>

<StackLayout Orientation="Vertical">

<StackLayout Orientation="Horizontal">

<Label Text="{Binding GroupId}" FontAttributes="Bold" Margin="20,20,0,0" HorizontalOptions="Start" VerticalOptions="Center" Scale="1.3"/>

<Label Text="{Binding NameOfGroup}" Margin="20,20,0,0" HorizontalOptions="StartAndExpand" VerticalOptions="CenterAndExpand"/>

</StackLayout>

</StackLayout>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

</ContentPage>

#### Файл Groups.xaml.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR.Views.Admin

{

public partial class Groups : ContentPage, INotifyPropertyChanged

{

public ObservableCollection<VKR.Models.Admin.Group> LG { get; private set; }

public Groups()

{

InitializeComponent();

this.OnAppearing();

BindingContext = this;

Routing.RegisterRoute("GroupCreate", typeof(VKR.Views.Admin.GroupCreate));

}

protected override void OnAppearing()

{

base.OnAppearing();

this.LG = App.DataBase.GetAdmin\_Group\_Data();

OnPropertyChanged("LG");

}

private async void AddToolbar\_Clicked(object sender, EventArgs e)

{

VKR.Models.Admin.Group g = new VKR.Models.Admin.Group();

await Navigation.PushAsync(new GroupCreate(ref g));

}

private async void collview\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (e.CurrentSelection != null)

{

VKR.Models.Admin.Group ww = e.CurrentSelection.FirstOrDefault() as VKR.Models.Admin.Group;

//Затычка на перевод в Лист со всеми редакциями!!!

await Navigation.PushAsync(new Group(ref ww));

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

}

}

#### Файл MapPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:ViewModels="clr-namespace:VKR.ViewModels.Admin"

x:Class="VKR.Views.Admin.MapPage"

xmlns:maps="clr-namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps"

x:DataType="ViewModels:MapPageViewModel"

Title="Карта"

xmlns:local="clr-namespace:VKR.ViewModels.Admin">

<ContentPage.Content>

<StackLayout>

<maps:Map

x:Name="MyMap"

IsShowingUser="true"

MapType="Street" />

<StackLayout Orientation="Horizontal" Margin="10">

<Button Text="Add pin"

Command="{Binding AddPin}"

HorizontalOptions="StartAndExpand"

x:Name="button1"/>

<Button Text="Change pin"

Command="{Binding ChangePin}"

HorizontalOptions="CenterAndExpand"/>

<Button Text="Remove pin"

Command="{Binding DeletePin}"

HorizontalOptions="EndAndExpand"/>

</StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

#### Файл MapPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using VKR.ViewModels.Admin;

namespace VKR.Views.Admin

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MapPage : ContentPage

{

public MapPage(ref VKR.Models.Admin.Group group)

{

InitializeComponent();

MapPageViewModel.map = MyMap;

MapPageViewModel.groupId = group.GroupId;

this.BindingContext = new MapPageViewModel();

}

}

}

#### Файл MarkerCreate.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Admin.MarkerCreate"

xmlns:local="clr-namespace:VKR.ViewModels.Admin">

<ContentPage.BindingContext>

<local:MarkerCreateViewModel />

</ContentPage.BindingContext>

<ContentPage.Content>

<StackLayout>

<Entry Placeholder="Название рабочей области" Text="{Binding MTitle}"/>

<Entry Placeholder="Примечания для рабочей области" Text="{Binding MText}"/>

<Entry Placeholder="радиус рабочей области" Text="{Binding MRadius}" Keyboard="Numeric"/>

<StackLayout Orientation="Horizontal" IsVisible="false" x:Name="mm">

<Label Text="Передвинуть маркер" HorizontalOptions ="StartAndExpand"/>

<Switch OnColor="Green" ThumbColor="LawnGreen" HorizontalOptions="EndAndExpand" IsToggled="{Binding MoveMarker}"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Button Text="Создать рабочую область" Command="{Binding CreateGeofence}" HorizontalOptions="StartAndExpand"/>

<Button Text="Вернуться назад" Command="{Binding GoBack}" HorizontalOptions="EndAndExpand"/>

</StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

#### Файл MarkerCreate.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using VKR.Models;

using VKR.ViewModels.Admin;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR.Views.Admin

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MarkerCreate : ContentPage

{

public MarkerCreate()

{

InitializeComponent();

if (MapPageViewModel.buttonClickedType == ButtonClickedType.Change)

mm.IsVisible = true;

}

}

}

#### Файл TimeCreate.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Admin.TimeCreate">

<ContentPage.Content>

<StackLayout Margin="20,0">

<Picker Title="День недели:" x:Name="picker">

<Picker.Items>

<x:String>Воскресенье</x:String>

<x:String>Понедельник</x:String>

<x:String>Вторник</x:String>

<x:String>Среда</x:String>

<x:String>Четверг</x:String>

<x:String>Пятница</x:String>

<x:String>Суббота</x:String>

</Picker.Items>

</Picker>

<StackLayout Orientation="Horizontal" Spacing="10">

<Label Text="Время начала" FontSize="Large"/>

<TimePicker Time="{Binding time\_start}" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<StackLayout Orientation="Horizontal" Spacing="10">

<Label Text="Время окончания" FontSize="Large"/>

<TimePicker Time="{Binding time\_end}" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<Button Text="Сохранить" Clicked="SaveButtonClicked"/>

<Button Text="Удалить" Clicked="DeleteButtonClicked" x:Name="deleteb"/>

</StackLayout>

</ContentPage.Content>

</ContentPage>

#### Файл TimeCreate.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR.Views.Admin

{

public partial class TimeCreate : ContentPage

{

VKR.Models.Admin.Shedule shedule;

int GroupId;

public TimeCreate(ref VKR.Models.Admin.Shedule shedule, int GroupId)

{

this.shedule = shedule;

this.GroupId = GroupId;

InitializeComponent();

picker.SetBinding(ItemsView.ItemsSourceProperty, nameof(VKR.Models.Admin.WeekDay));

if (shedule.id == -1)

deleteb.IsEnabled = deleteb.IsVisible = false;

else

picker.SelectedIndex = ((int)this.shedule.weekDay);

}

protected override void OnAppearing()

{

base.OnAppearing();

this.BindingContext = shedule;

}

private async void SaveButtonClicked(object sender, EventArgs e)

{

if (picker.SelectedIndex != -1)

{

shedule.weekDay = (Models.Admin.WeekDay)picker.SelectedIndex;

App.DataBase.SaveShedule(shedule, GroupId);

await Navigation.PopAsync();

}

else

await DisplayAlert("Пустые поля!", "Заполните поля, прежде чем сохранять!", "Ок");

}

private async void DeleteButtonClicked(object sender, EventArgs e)

{

var ans = await DisplayAlert("Удаление расписания", "Удалить данное расписание?", "Да", "Нет");

if (ans == true)

{

App.DataBase.DeleteShedule(shedule, GroupId);

await Navigation.PopAsync();

}

}

}

}

#### Файл TimeShedule.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Admin.TimeShedule"

Title="Расписание">

<StackLayout Orientation="Vertical" Spacing="5">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="125" />

<ColumnDefinition Width="125" />

<ColumnDefinition Width="125" />

</Grid.ColumnDefinitions>

<Label Text="День"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="1"

Text="Начало"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="2"

Text="Конец"

HorizontalOptions="Center"

VerticalOptions="Center" />

</Grid>

<CollectionView SelectionMode="Single"

SelectionChanged="CollectionView\_SelectionChanged"

x:Name="collView">

<CollectionView.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="125" />

<ColumnDefinition Width="125" />

<ColumnDefinition Width="125" />

</Grid.ColumnDefinitions>

<Label Text="{Binding weekDay}"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="1"

Text="{Binding time\_start}"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="2"

Text="{Binding time\_end}"

HorizontalOptions="Center"

VerticalOptions="Center" />

</Grid>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

<Button Text="Добавить" Clicked="AddToolbar\_Clicked" VerticalOptions="End"/>

</StackLayout>

</ContentPage>

#### Файл TimeShedule.xaml.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using VKR.Models.Admin;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace VKR.Views.Admin

{

public partial class TimeShedule : ContentPage, INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

private VKR.Models.Admin.Group \_group

{

get; set;

}

public VKR.Models.Admin.Group group

{

get

{

return \_group;

}

set

{

\_group = value;

OnPropertyChanged("LS");

OnPropertyChanged("group");

}

}

public TimeShedule(ref VKR.Models.Admin.Group group)

{

group.LS = App.DataBase.GetShedules(group.GroupId);

this.group = group;

InitializeComponent();

collView.SetBinding(ItemsView.ItemsSourceProperty, nameof(this.group.LS));

}

protected override void OnAppearing()

{

base.OnAppearing();

group.LS = App.DataBase.GetShedules(group.GroupId);

this.BindingContext = group;

}

private async void AddToolbar\_Clicked(object sender, EventArgs e)

{

VKR.Models.Admin.Shedule s = new VKR.Models.Admin.Shedule();

await Navigation.PushAsync(new TimeCreate(ref s, group.GroupId));

}

private async void CollectionView\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (e.CurrentSelection != null)

{

VKR.Models.Admin.Shedule shedule = e.CurrentSelection.FirstOrDefault() as VKR.Models.Admin.Shedule;

await Navigation.PushAsync(new TimeCreate(ref shedule, group.GroupId));

}

}

}

}

#### Файл AllData.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Watcher.AllData">

<CollectionView SelectionMode="Single"

SelectionChanged="OnCollectionViewSelectionChanged"

x:Name="collview">

<CollectionView.ItemTemplate>

<DataTemplate>

<StackLayout Orientation="Vertical"

Spacing="5">

<StackLayout Orientation="Horizontal">

<Label Text="ФИО:" Margin="20,0,0,0"/>

<Label Text="{Binding FIO}" FontAttributes="Bold" Margin="0,0,20,0" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="Должность" Margin="20,0,0,0"/>

<Label Text="{Binding Position}" Margin="0,0,20,0" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<StackLayout Orientation="Horizontal" MinimumHeightRequest="200">

<Label Text="Рабочая группа:" Margin="20,0,0,0" MinimumWidthRequest="200" WidthRequest="120"/>

<Label Text="{Binding WorkGroup}" Margin="0,0,20,0" HorizontalOptions="EndAndExpand" HorizontalTextAlignment="End"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="Количество пропусков:" Margin="20,0,0,0"/>

<Label Text="{Binding NumberOfAbsence}" Margin="0,0,20,0" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="Время пропусков:" Margin="20,0,0,0"/>

<Label Text="{Binding SumTimeOfAbsence}" Margin="0,0,20,50" HorizontalOptions="EndAndExpand"/>

</StackLayout>

</StackLayout>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

</ContentPage>

#### Файл AllData.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using Newtonsoft.Json;

using VKR.Models.Watcher;

using VKR.Data;

using System.Collections.ObjectModel;

using System.ComponentModel;

namespace VKR.Views.Watcher

{

public partial class AllData : ContentPage, INotifyPropertyChanged

{

public ObservableCollection<WatchWorker> Lww { get; set; }

public event PropertyChangedEventHandler PropertyChanged;

private async void OnCollectionViewSelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (e.CurrentSelection != null)

{

WatchWorker ww = e.CurrentSelection.FirstOrDefault() as WatchWorker;

/\*string jason = await Task.Run(() => JsonConvert.SerializeObject(new AboutWorker(ww.id, ww.FIO, ww.Position, ww.WorkGroup, ww.NumberOfAbsence, TimeSpan.Parse(ww.SumTimeOfAbsence))));\*/

await Navigation.PushAsync(new PersonData(ww));

}

}

protected override void OnAppearing()

{

base.OnAppearing();

Lww = App.DataBase.GetWatcher\_Data();

OnPropertyChanged("Lww");

}

public AllData()

{

InitializeComponent();

collview.SetBinding(ItemsView.ItemsSourceProperty, nameof(Lww), BindingMode.TwoWay);

this.OnAppearing();

BindingContext = this;

}

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

}

}

#### Файл PersonData.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VKR.Views.Watcher.PersonData">

<ContentPage.Content>

<StackLayout Orientation="Vertical"

Spacing="5">

<StackLayout Orientation="Horizontal">

<Label Text="ФИО:" Margin="20,0,0,0"/>

<Label Text="{Binding FIO}" FontAttributes="Bold" Margin="0,0,20,0" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="Должность" Margin="20,0,0,0"/>

<Label Text="{Binding Position}" Margin="0,0,20,0" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<StackLayout Orientation="Horizontal" MinimumHeightRequest="200">

<Label Text="Рабочая группа:" Margin="20,0,0,0" MinimumWidthRequest="200" WidthRequest="120"/>

<Label Text="{Binding WorkGroup}" Margin="0,0,20,0" HorizontalOptions="EndAndExpand" HorizontalTextAlignment="End"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="Количество пропусков:" Margin="20,0,0,0"/>

<Label Text="{Binding NumberOfAbsence}" Margin="0,0,20,0" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="Время пропусков:" Margin="20,0,0,0"/>

<Label Text="{Binding SumTimeOfAbsence}" Margin="0,0,20,0" HorizontalOptions="EndAndExpand"/>

</StackLayout>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="95" />

<ColumnDefinition Width="95" />

<ColumnDefinition Width="95" />

<ColumnDefinition Width="95" />

</Grid.ColumnDefinitions>

<Label Text="Дата"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="1"

Text="Начало"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="2"

Text="Конец"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="3"

Text="Итого"

HorizontalOptions="Center"

VerticalOptions="Center" />

</Grid>

<CollectionView SelectionMode="Single"

SelectionChanged="CollectionView\_SelectionChanged"

x:Name="collView">

<CollectionView.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="40" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="95" />

<ColumnDefinition Width="95" />

<ColumnDefinition Width="95" />

<ColumnDefinition Width="95" />

</Grid.ColumnDefinitions>

<Label Text="{Binding date}"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="1"

Text="{Binding start\_time}"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="2"

Text="{Binding end\_time}"

HorizontalOptions="Center"

VerticalOptions="Center" />

<Label Grid.Column ="3"

Text="{Binding skip\_time}"

HorizontalOptions="Center"

VerticalOptions="Center" />

</Grid>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

#### Файл PersonData.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using VKR.Models.Watcher;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using Newtonsoft.Json;

using System.Collections.ObjectModel;

using System.ComponentModel;

namespace VKR.Views.Watcher

{

public partial class PersonData : ContentPage, INotifyPropertyChanged

{

protected override void OnAppearing()

{

base.OnAppearing();

this.BindingContext = aboutWorker;

}

public event PropertyChangedEventHandler PropertyChanged;

private WatchWorker AboutWorker

{

get; set;

}

public WatchWorker aboutWorker {

get

{

return AboutWorker;

}

set

{

AboutWorker = value;

OnPropertyChanged("aboutWorker");

OnPropertyChanged("FIO");

OnPropertyChanged("Position");

OnPropertyChanged("WorkGroup");

OnPropertyChanged("NumberOfAbsence");

OnPropertyChanged("SumTimeOfAbsence");

}

}

public PersonData(WatchWorker aw)

{

aboutWorker = aw;

InitializeComponent();

collView.SetBinding(ItemsView.ItemsSourceProperty, nameof(aboutWorker.skip\_list));

}

private async void CollectionView\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var ans = await DisplayAlert("Удаление записи", "Вы уверены, что данный пропуск был совершен на законных причинах", "Да", "Нет");

if (ans == true)

{

if (e.CurrentSelection != null)

{

Worker\_skip ws = e.CurrentSelection.FirstOrDefault() as Worker\_skip;

if (App.DataBase.DeleteWTS(ws.skip\_id))

{

aboutWorker.NumberOfAbsence--;

TimeSpan ts = TimeSpan.Parse(aboutWorker.SumTimeOfAbsence);

ts = ts.Subtract(TimeSpan.Parse(ws.skip\_time));

aboutWorker.SumTimeOfAbsence = ts.ToString();

aboutWorker.skip\_list.Remove(ws);

OnPropertyChanged("aboutWorker");

}

else

await DisplayAlert("Удаление записи", "Не удалось подключится к БД для удаления пропуска", "Ок");

}

}

}

protected void OnPropertyChanged(string propName)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(propName));

}

}

}