Введение.

В современном мире существует уже более 2,5 млрд. автомобилей различных марок и моделей. В следствии чего возникает большой спрос на ремонт и обслуживание. Но не все люди готовы тратить большие деньги на ремонт, тем более если этот ремонт можно выполнить самостоятельно. При самостоятельном ремонте у человека может возникать большое количество вопросов из-за незнания технического устройства узлов и агрегатов автомобиля, в следствии чего человек прибегает к помощи интернета. Но, к сожалению, на просторах интернета не всегда можно найти нужную правдивую информацию.

Приложение «LADA Helper» - это приложение помощник при ремонте автомобиля. Разрабатываемое приложение должно решить проблему, описанную ранее. Например, у человека произошла поломка автомобиля, но по определённым причинам он не может посетить автосервис или желает произвести ремонт самостоятельно. Чтобы сделать ремонт быстро и качественно, он может воспользоваться приложением, в котором будет полная и достоверная информация о ремонте, а также видео сопровождение в качестве наглядного примера. Более того, приложение так же будет полезно для специалистов автомобильного ремонта, так как приложение может быть хорошим ассистентом и методическим пособием.

В качестве дополнения, как ассистент, в приложении можно вести журнал замены масла, а также найти сервис для обслуживания своего автомобиля, в случае, если владелец не может произвести ремонт самостоятельно.

1.Техническое задание

Как уже говорилось ранее, приложение является помощником при техническом обслуживании автомобиля. На сегодняшний день существует различные варианты подобных приложений у новых компаний, недавно пришедших на российский рынок. Но для отечественного автопрома качественного приложения не существует до сих пор – это факт побуждает большой интерес для разработчиков. Главной задачей этой работы является создание новой информационной системы (приложения) для ремонта отечественных автомобилей.

Если говорить о самом приложении, то функционал требует быть достаточно обширным для удовлетворения большинства потребностей пользователя. В следствии чего можно выделить несколько основных пунктов функционала приложения:

1) При запуске приложения пользователю будет необходимо зарегистрироваться в системе или же авторизоваться, если пользователь уже был зарегистрирован ранее. Информация о паролях и почте пользователя должна быть записана в базе данных, куда будет ссылаться система авторизации.

2) Основным функционалом приложения является помощь в ремонте и обслуживании автомобиля, это значит, что необходимо создать выбор конкретной модели автомобиля и обеспечить удобный поиск информации о поломке. Страница информации о поломки должна содержать информацию о поломке «Симптомы» поломки», а также подробную информацию о возможностях решения данной поломки. В качестве наглядного примера должно быть видео сопровождение (интегрированное с площадки YouTube в целях экономии занимаемого места приложением на устройствах пользователя).

3) Каждый автовладелец несколько раз в год производит замену моторного масла. Как правило, регламент указывает на отметку в 8-10 тысяч километров до следующей замены. Многие забывают, когда в последний раз была произведена замена масла и не ведут записи о произошедшем техническом обслуживании. Приложение должно решать и эту проблему: на вкладке «Мои записи» находится журнал, в котором пользователь может оставлять заметки об обслуживании своего автомобиля.

1. В целях удобства пользователей на вкладке «Поиск сервисов» будет находится справочная информация об официальных сервисах, производящих техническое обслуживание автомобилей. В качестве составляющих справочной информации, будут находиться следующие поля: название сервиса, адрес, контактная информация и часы работы.

Требования при создании информационной системы можно разделить на 2 вида: Функциональные и Нефункциональные. Функциональные требования. Это перечень сервисов, которые должна выполнять система, причем должно быть указано, как система реагирует на те или иные вход­ные данные, как она ведет себя в определенных ситуациях и т.д. В некоторых слу­чаях указывается, что система не должна делать. Нефункциональные требования. Описывают характеристики системы и ее окружения, а не поведение системы. Здесь также может быть приведен перечень ограничений, накладываемых на действия и функции, выполняемые системой. Они включают временные ограничения, ограничения на процесс разработки системы, стандарты и т.д.

В действительности четкой границы между этими типами требований не существует. Например, пользовательские требования, касающиеся безопасности системы, можно отнести к нефункциональным. Однако при более детальном рассмотрении такое требование можно отнести к функциональным, поскольку оно порождает необходимость включения в систему средства авторизации пользователя. Поэтому, рассматривая далее эти виды требований, мы должны всегда помнить, что данная классификация в значительной степени искусственна.

* 1. Функциональные требования

1) Система регистрации и авторизации

2) Предоставление информации по техническому обслуживанию и ремонту автомобиля.

3) Возможность вести записи по техническому обслуживанию и ремонту (на примере замены масла).

4) Предоставление пользователю информации о дилерских сервисах для ремонта автомобилей.

1.2 Нефункциональные требования

1)Стабильная работа приложения (минимальное количество ошибок и сбоев).

2)Адаптивность под наибольшее количество версий Android т.е. 95% начиная с версии Android 5.0(Lollipop).

3) быстрый поиск информации (время поиска не должно превышать более 1 минуты).

5)Интуитивно понятный и простой интерфейс.

2.Проектирование.

StarUML ™ предполагает ясное понимание концептуального различия между моделями, представлениями и диаграммами. Модель - элемент, который содержит всю информацию о модели программы. Представление - визуальное выражение информации, содержавшейся в модели, а Диаграмма - коллекция визуальных образов, которая отображает определенные аспекты проекта.

2.1 Диаграмма прецедентов.

Поведение системы, иными словами функциональность, которую она обеспечивает описывают с помощью функциональной модели, которая отображает системные прецеденты (use cases, случаи использования), системное окружение (действующих лиц, актеров, actors) и связи между ними (use cases diagrams). Диаграмма прецедентов отображена на Рис.1.

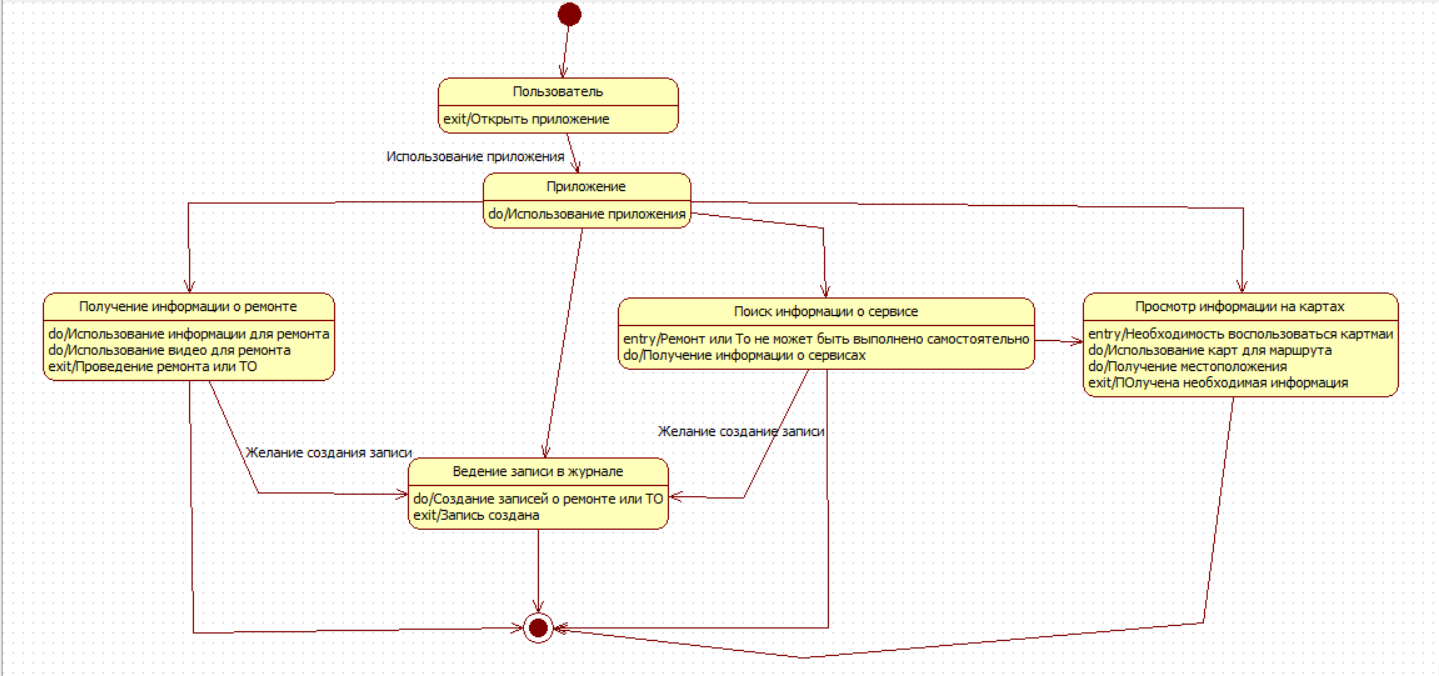
 Рис.1 Диаграмма прецедентов

На диаграмме прецедентов отображены два актора: пользователь и автомобиль. Пользователь имеет ассоциацию с автомобилем, который в свою очередь ассоциируется с поломками. Также пользователь имеет ассоциацию с приложением помощником, поскольку является основным пользователем приложения. Система регистрации и авторизации, информация о поломке, журнал записей пользователя, карты и информационный справочник отмечены как включающиеся составляющие самого приложения. Полученная информация о пользователе и информация о логинах и паролях отмечены как расширения системы регистрации и авторизации. Список работ для ремонта и видео отмечены как расширение информации о поломке.

Записи отмечены как расширение для системы записей о ремонте автомобиля. Контактная информация, адрес и название сервиса отмечены как расширения информационного справочника. Местоположение пользователя и просмотр маршрутов, а также просмотр пробок отмечены как расширение для карт. Эта диаграмма позволяет понять, как работает функционал приложения. Пользователь, имеющий автомобиль, который сломался, может воспользоваться приложением, которое поможет ему с ремонтом. В приложении он может получить информацию о ремонте, сделать записи, а также найти подходящий сервис в случае, если произвести ремонт самостоятельно не представляется возможным, более того у пользователя есть возможность использования карт, при этом ему ненужно использовать сторонние приложения.

2.2 Диаграмма состояний.

**Диаграмма состояний (State diagram)** — это один из видов диаграмм UML, используемых в разработке программного обеспечения, чтобы визуализировать и моделировать поведение объекта или системы в различных состояниях. Она позволяет описать все возможные состояния объекта, а также переходы между ними в ответ на определенные события. Диаграмма состояний описывает все возможные состояния, в которых может находиться объект или система, а также переходы между этими состояниями. Она позволяет визуально представить различные состояния объекта или системы и показать, как они изменяются в ответ на события или внешние условия. Переходы обозначают изменение состояния и указывают, при каких условиях происходит переход между состояниями. События являются внешними сигналами или действиями, которые вызывают переходы между состояниями. Диаграмма состояний приведена на Рис.2.

 Рис.2 Диаграмма состояний.

На данной диаграмме отображены состояния, которые требуют определённых действий, для перехода в следующее состояние. В случае разрабатываемой системы, при наличии проблем у пользователя с автомобилем, он откроет приложение где ему будет необходимо зарегистрироваться или же авторизоваться. При успешной регистрации и авторизации будет получен доступ к функционалу.

2.3 Диаграмма активностей.

Диаграмма активностей (видов деятельности) отображает динамические аспекты поведения системы. Эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой. Диаграмма показана на Рис.3



Рис.3 Диаграмма активности.

Диаграмма активностей отображает возможные варианты использования системы. В данном случае на схеме указаны пять полей (сторон) на которых происходят действия: Автомобиль, включающий в себя поломку или необходимость ТО, Пользователь, Система получения информации о ремонте, система записей и Система получения информации о сервисах. На примере этой диаграммы можно привести один из случаев использования. Например, у пользователя появилась необходимость замены масла он открывает приложение и ищет информацию о том, как можно самостоятельно поменять масло в своём автомобиле. Далее он понимает, что сделать данную работу он может самостоятельно и сам меняет масло. Чтобы не забыть когда он последний раз проводил техническое обслуживание, он делает запись в журнале технических обслуживаний с указанием даты пробега и марки масла. Иным же примером может быть случай, когда автовладелец не может самостоятельно провести ремонтные работы, в таком случае у него есть возможность воспользоваться справочником, для поиска сервисов где можно провести обслуживание.

2.4 Диаграмма классов

Диаграмма классов – это UML-диаграмма, которая описывает систему, визуализируя различные типы объектов внутри системы и виды статических связей, которые существуют между ними. Он также иллюстрирует операции и атрибуты классов. Диаграмма показана на Рис.4.

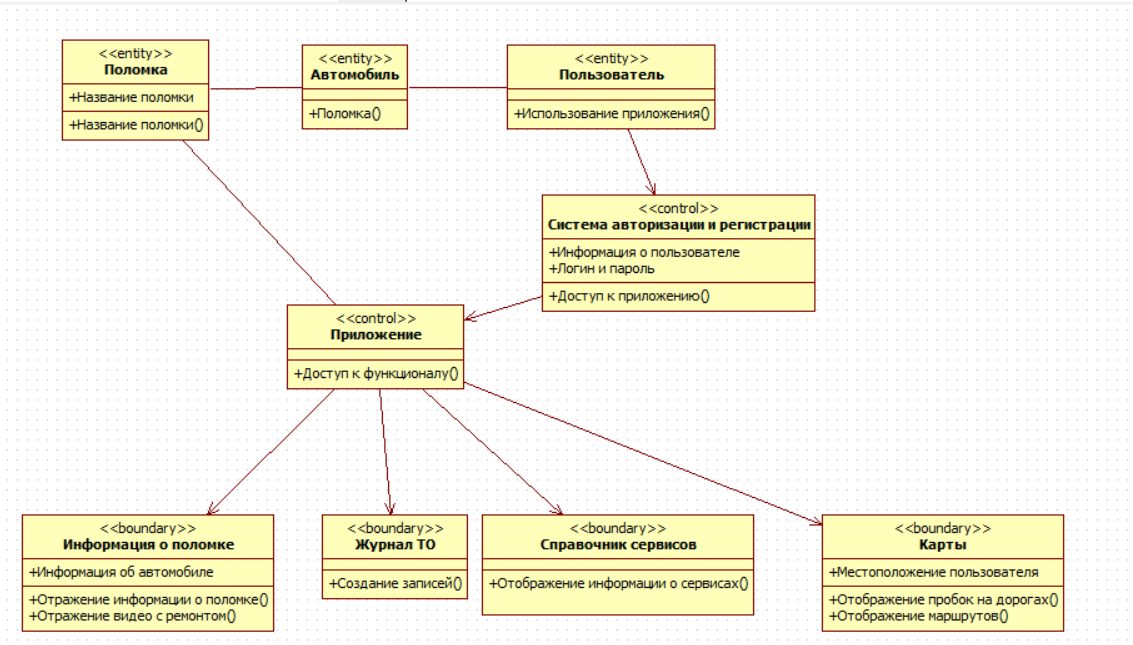


Рис.4 Диаграмма классов.

На диаграмме классов отображены взаимосвязи между выявленными классами. Более того можно заметить, что классы разделены по типам: Поломка, автомобиль и пользователь объявлены как классы-сущности; Само приложение и система авторизации и регистрации объявлены как контролирующие классы, с помощью которого и происходит основное действие по поиску информации; Информация о поломке, Карты, журнал записей и справочник сервисов объявлены как граничные классы поскольку они взаимодействуют между пользователем и системой.

2.5 Диаграмма последовательности.

Диаграмма последовательности — это подвид диаграмм взаимодействия, который позволяет описать взаимодействие между объектами в системе в виде последовательности сообщений, действий и операций, отображая порядок выполнения действий и обмена информацией между объектами во времени.

Диаграммы последовательности отображают взаимодействия между объектами, порядок выполнения операций и обмена информацией во времени. Они помогают улучшить понимание функционирования системы, выявить потенциальные ошибки и проблемы в процессе взаимодействия объектов.

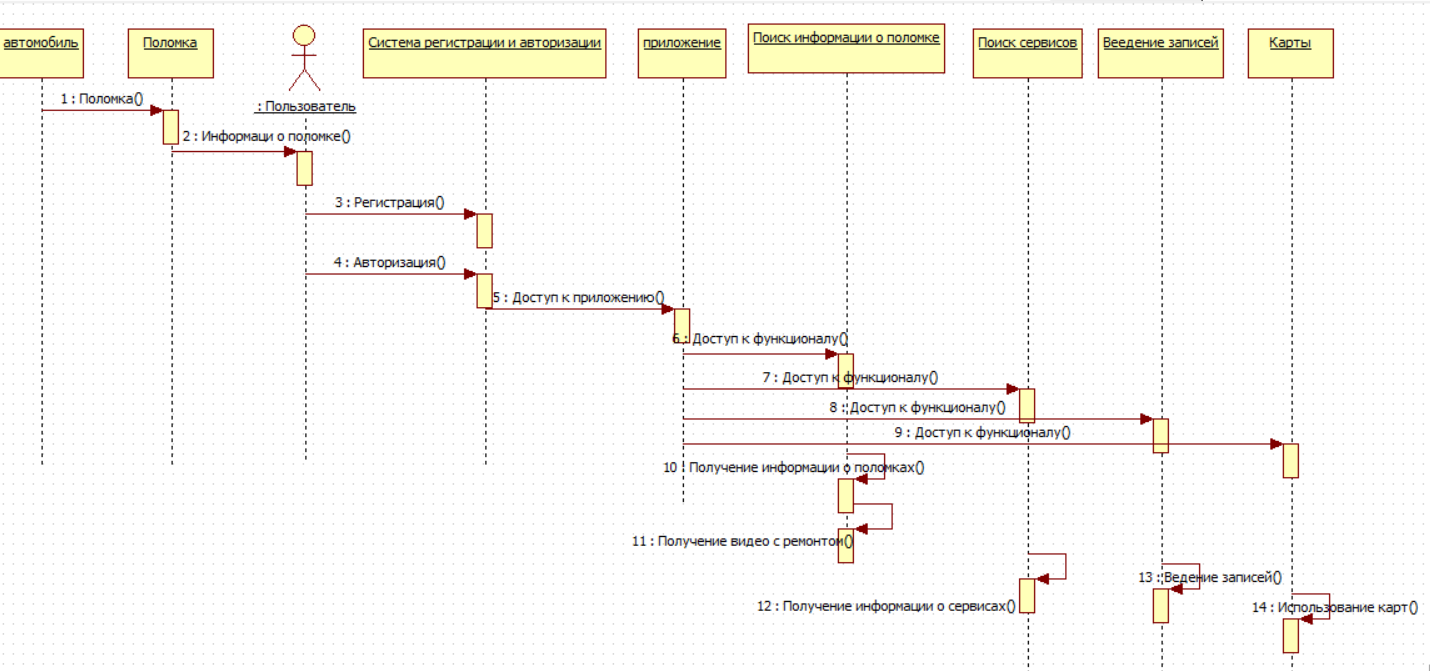


Рис.5 Диаграмма последовательности.

2.6 Диаграмма развертывания

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы. Используя его, вы можете понять, как система будет физически развернута на аппаратном обеспечении.

Диаграммы развертывания помогают моделировать аппаратную топологию системы по сравнению с другими типами UML-диаграмм, которые в основном описывают логические компоненты системы. Диаграмма развёртывания отображена на Рис.6.

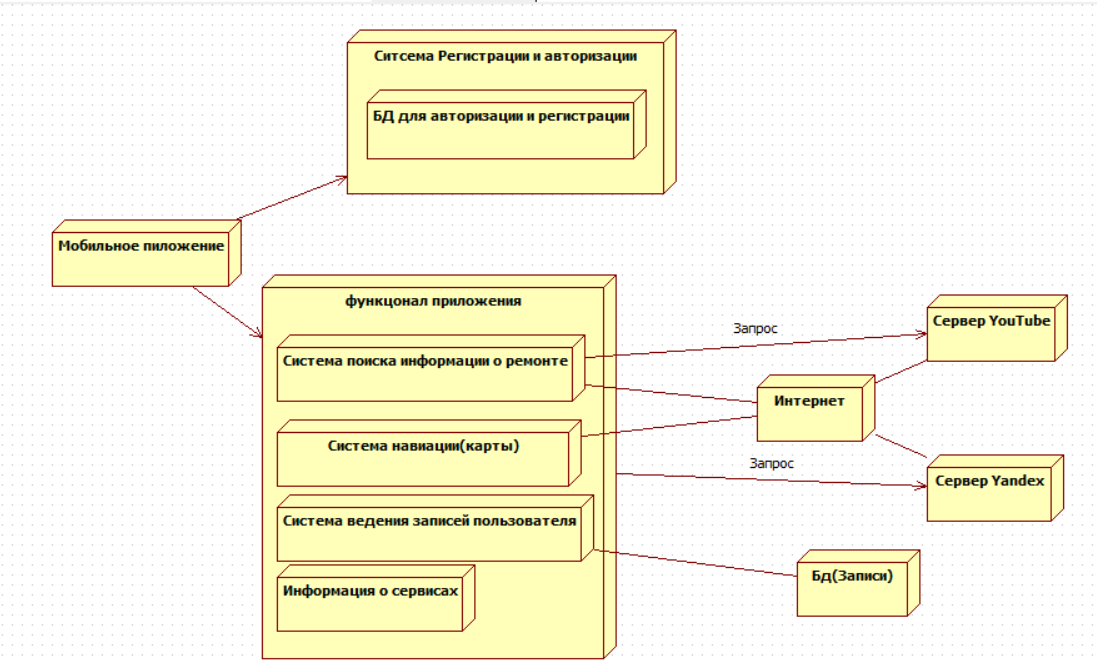


Рис.6 Диаграмма развёртывания

2.7. Диаграмма Компонентов

Диаграмма компонентов описывает особенности физического представления системы а так же описывает следующие задачи:

1)визуализация общей структуры исходного кода программной системы

2)спецификация исполнимого варианта программной системы

3)обеспечение многократного использования отдельных фрагментов программного кода

4)представление концептуальной и физической схем баз данных

Диаграмма компонентов показана на Рис.7.

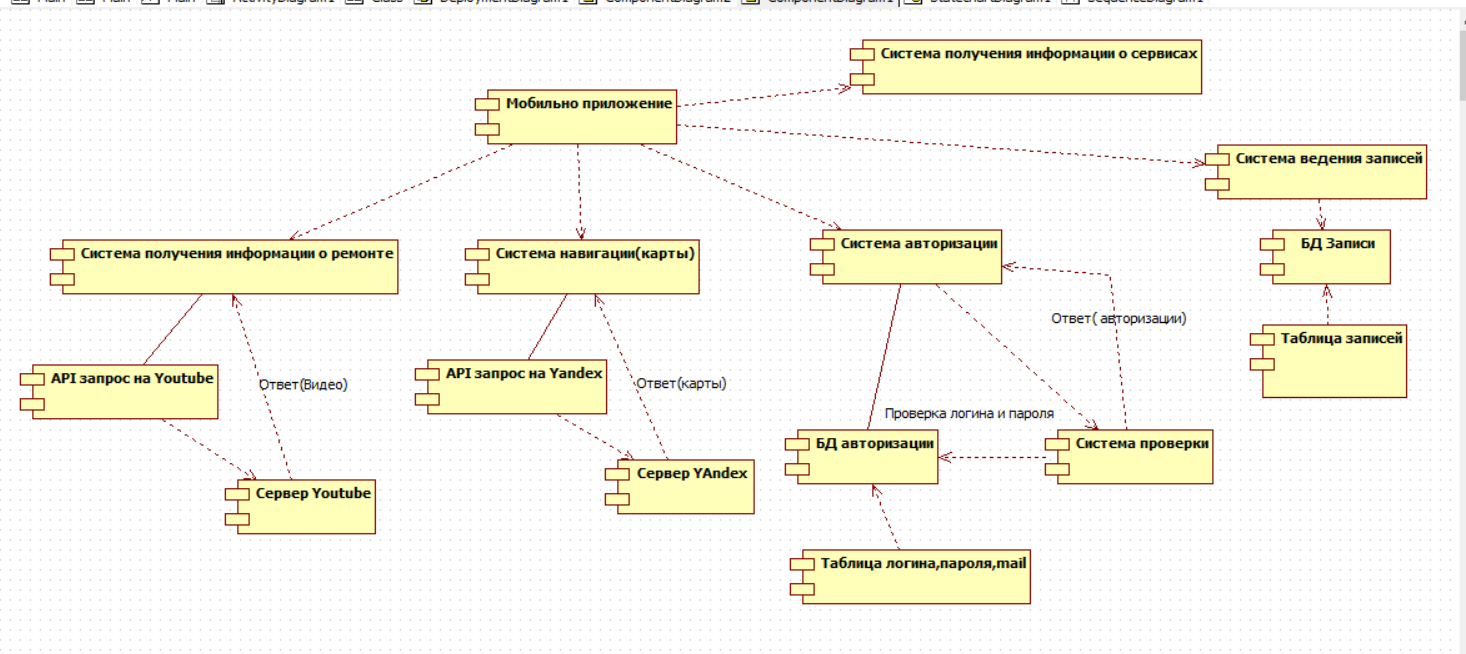
****

Рис.7 Диаграмма компонентов.

2.8 Схема БД

Схема базы данных представляет собой логическую конфигурацию либо целой реляционной базы данных, либо ее части. Схема может существовать как в виде наглядного представления базы данных, так и в виде набора формул (также именуемых «условиями целостности»), которые регулируют ее устройство. Эти формулы выражаются с помощью языка описания данных, например, SQL. Будучи частью словаря данных, схема показывает, как связаны между собой сущности, из которых состоит база данных (таблицы, представления, хранимые процедуры и так далее).

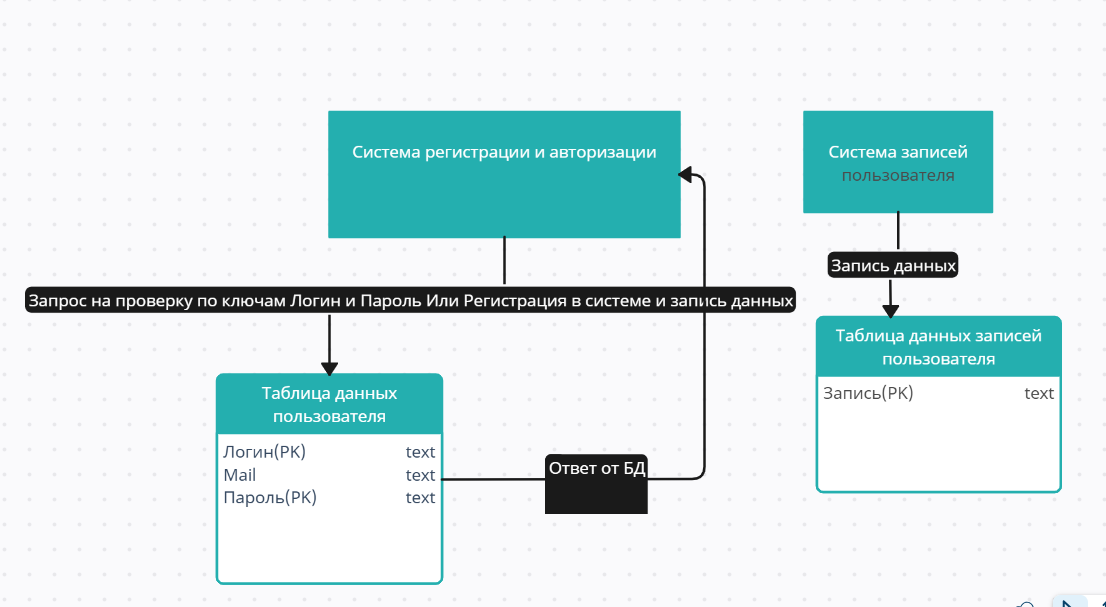


Рис.8 Схема работы БД

3.Тест кейсы к БД (на регистрацию и авторизацию)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер | Название | Требование | Валидные данные | Невалидные данные | Результат | Результат работы. |
| 1 | Регистрация пользователя | Логин:  не менее 6 символов  Пароль:  не менее 6 символов  Mail.  Заполнены все поля | Yura  Yuryi220  Yuryi220@mail.ru | -  -  - | Регистрация не выполнена | Ошибка заполнения полей: логин менее 6 символов |
| 2 | Регистрация пользователя | Логин:  не менее 6 символов  Пароль:  не менее 6 символов  Mail.  Заполнены все поля | YuraLitvin  Y1Our  Yuryi220@mail.ru | -  -  - | Регистрация не выполнена | Ошибка заполнения полей: пароль  менее 6 символов |
| 3 | Регистрация пользователя | Логин:  не менее 6 символов  Пароль:  не менее 6 символов  Mail.  Заполнены все поля | YuraLitvin  Y1Our221 | -  -  - | Регистрация не выполнена | Ошибка заполнения полей: Не все поля заполнены |
| 4 | Регистрация пользователя | Логин:  не менее 6 символов  Пароль:  не менее 6 символов  Mail.  Заполнены все поля | YuraLitvin  Y1Our221  Yuryi220@mail.ru | -  -  - | Регистрация прошла успешно | Ошибок нет |
| 5 | Авторизация пользователя | Логин:  не менее 6 символов  Пароль:  не менее 6 символов | YuraLitvin  Y1Our221 | YuraLitv  Y1Our221 | Авторизация не выполнена | Ошибка: пользователь не найден |
| 6 | Авторизация пользователя | Логин:  не менее 6 символов  Пароль:  не менее 6 символов | YuraLitvin  Y1Our221 | YuraLitvin  Y1Our1 | Авторизация не выполнена | Ошибка: Неверный пароль |
| 7 | Авторизация пользователя | Логин:  не менее 6 символов  Пароль:  не менее 6 символов | YuraLitvin  Y1Our221 | YuraLitvin  - | Авторизация не выполнена | Ошибка:  Заполнены не все поля |
| 8 | Авторизация пользователя | Логин:  не менее 6 символов  Пароль:  не менее 6 символов | YuraLitvin  Y1Our221 | YuraLitvin  Y1Our221 | Авторизация выполнена | Ошибок нет |

5.Реализация

Как уже говорилось ранее при запуске приложения пользователя встречает система регистрации и авторизации. Система регистрации отображена на Рис.4

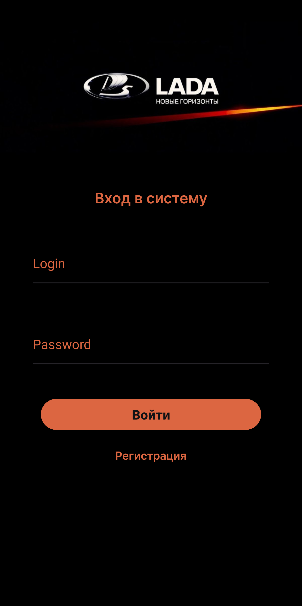
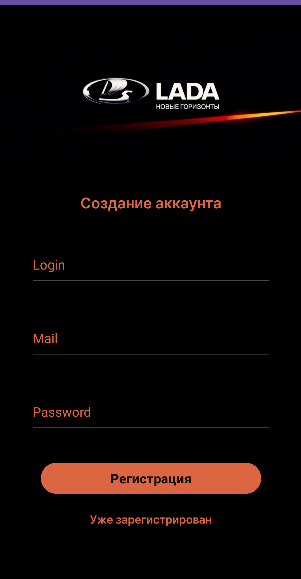


Рис.9 Система регистрации и авторизации.

На странице создания аккаунта пользователю необходимо заполнить такие поля как логин, email и пароль, после чего необходимо нажать кнопку «Регистрация». После нажатия на кнопку будет создан объект User, а все данные будут записаны в базу данных «DBU» в таблицу users по соответствующим ключам с помощью вызова функции addUser.



Рис.5 Создание базы данных при помощи SQL команды

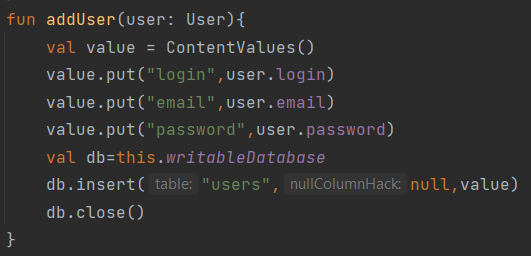


Рис.6 Функция добавления данных в БД.

Когда пользователь зарегистрировался или в случае, если пользователь уже был зарегистрирован он может перейти на страницу авторизации, нажав на кнопку «Уже зарегистрирован». Таким образом пользователь попадёт на страницу авторизации, где ему будет необходимо заполнить поля Логина и Пароля для авторизации. Система авторизации после нажатия на кнопку «Войти» производит поиск данных в БД при помощи SQL команды, вызвав функцию CheckUs.

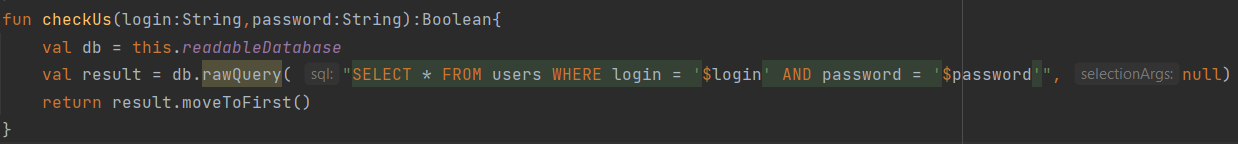


Рис.7 поиск пользователя в базе данных при вызове функции CheckUs.

В случае если не все поля заполнены при авторизации или регистрации появляется системное уведомление «Заполните все поля». Если пользователь успешно зарегистрирован или авторизован, появляется системное уведомление «Успешно». В случае, если пользователь не был зарегистрирован ранее, то после нажатия на кнопку «Войти», появится системное уведомление «Пользователь не найден»

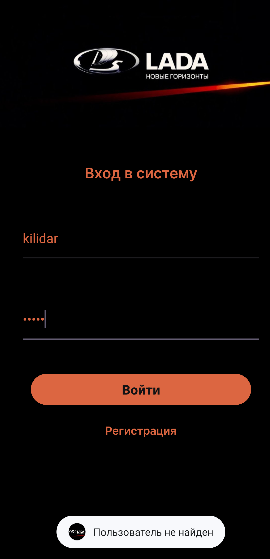


Рис.10 Пример системного уведомления.

После успешной регистрации и авторизации, пользователь попадает на главный экран, где может воспользоваться таким функционалом как «Выбор автомобиля», «Поиск сервиса» и «Мои записи». Главный экран показан на Рис.9.



Рис.11 Главная страница.

Чтобы получить информацию о ремонте, пользователю необходимо нажать на кнопку «Выбор автомобиля» после чего он перейдет на страницу выбора автомобилей, где он может выбрать интересующий его автомобиль. После выбора автомобиля пользователю откроется страница с перечнем поломок. На странице «Поломки» пользователю будет предоставлена информация о том, как самостоятельно произвести ремонт своего транспортного средства, а также в конце страницы выбранной поломки будет видео сопровождения, где в качестве наглядного примера продемонстрирован ремонт транспортного средства.

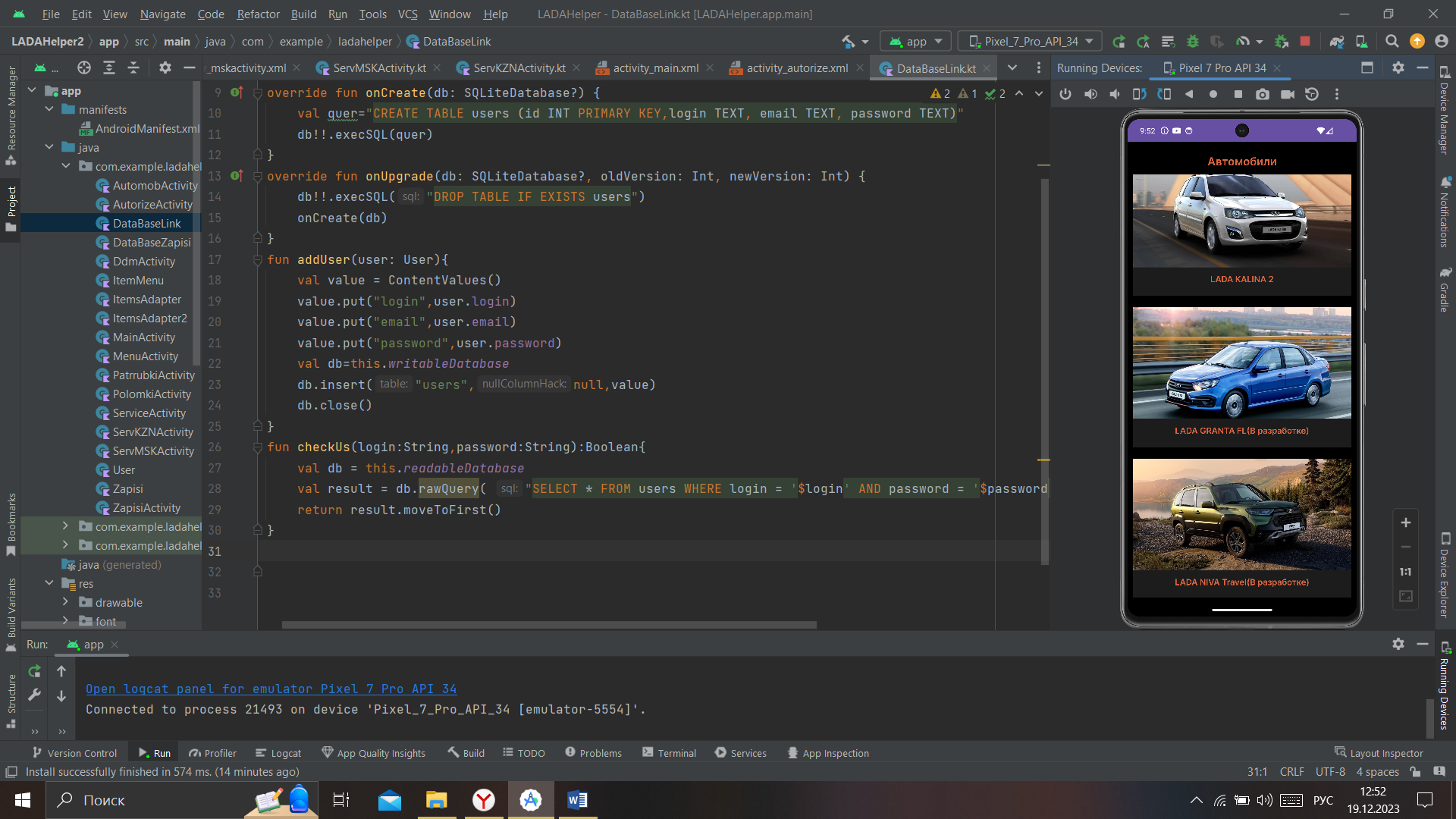


Рис.12 Страница выбора автомобилей.

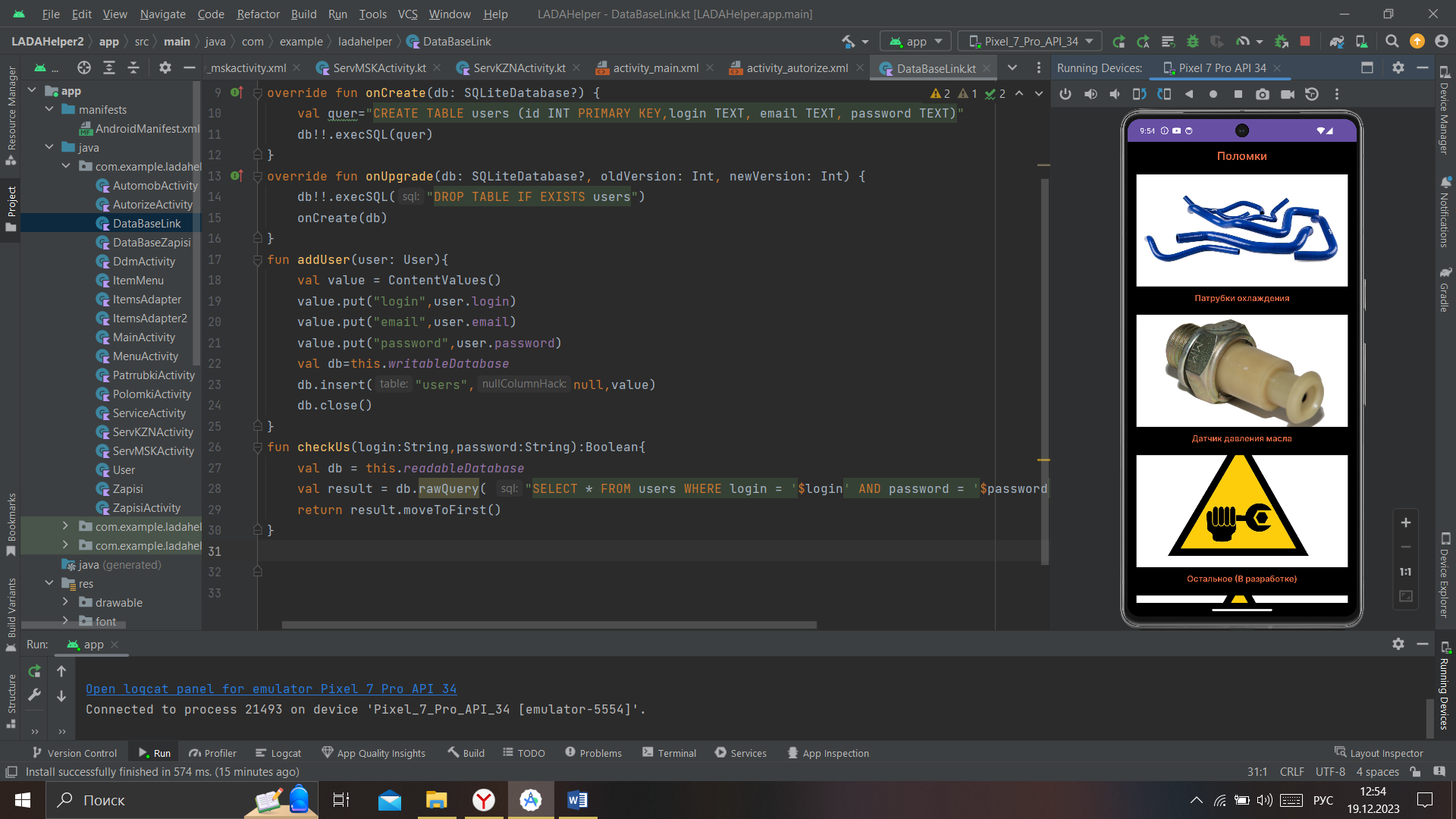


Рис.13 Страница выбора поломок для автомобиля LADA KALINA 2

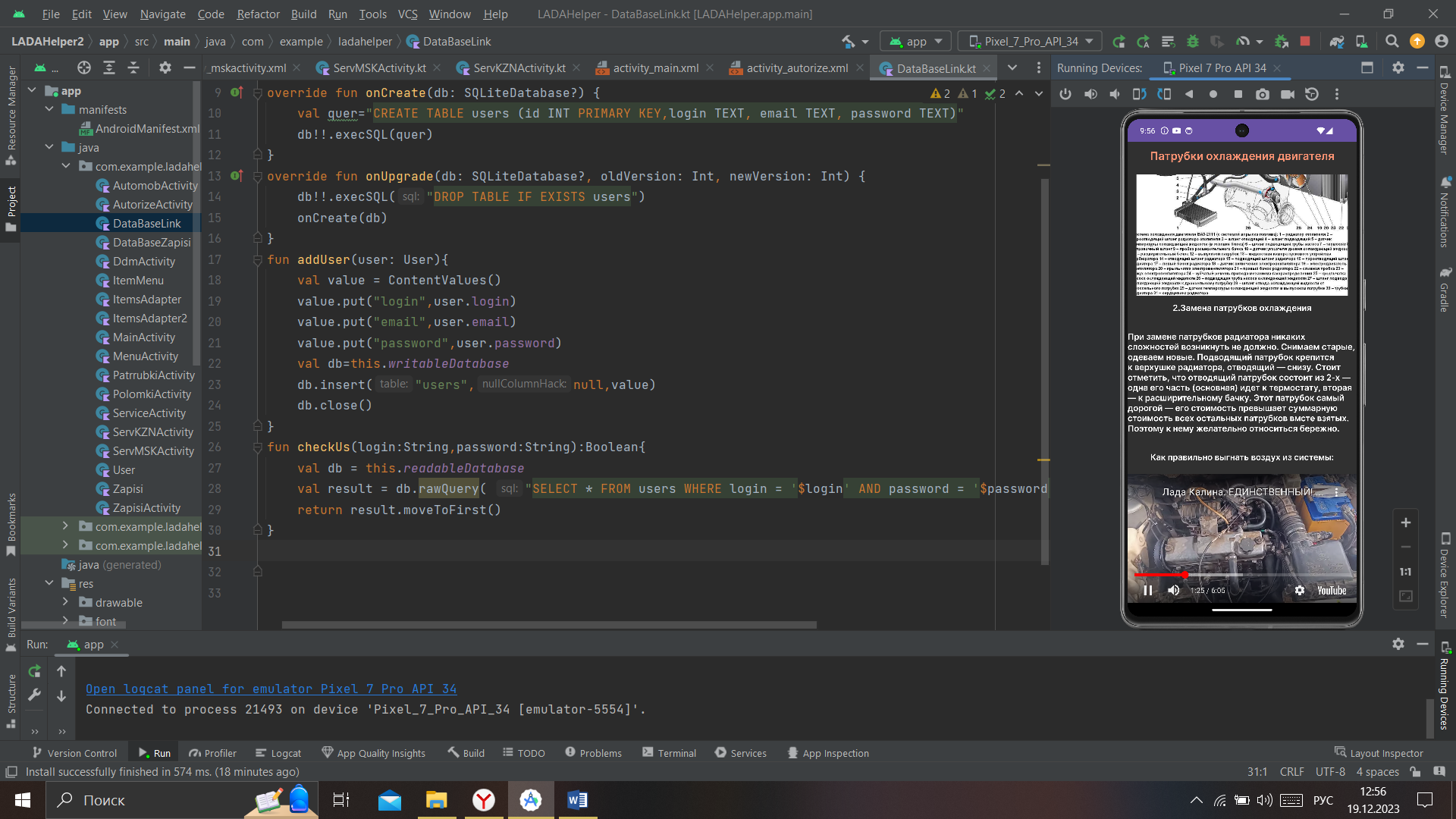
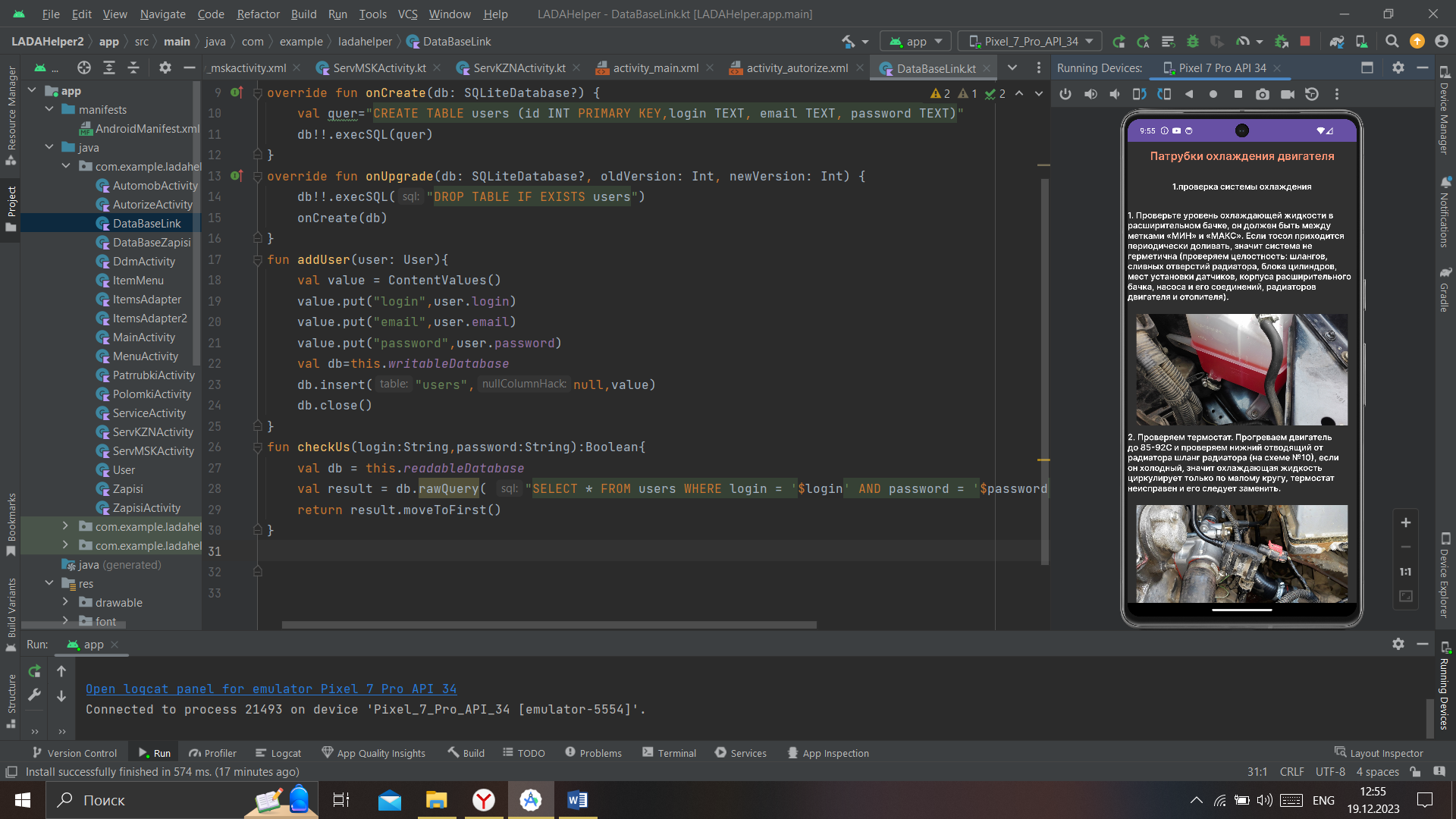


Рис.14 Страница поломки «Патрубки охлаждения»

Как видно на Рис.12 в конце страницы было интегрировано видео с платформы YouTube в целях экономии веса приложения. Для интеграции видео в классе PatrubkiActivity этой страницы был написан следующий код:

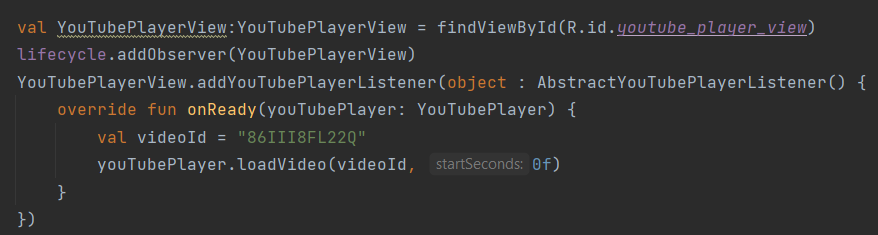


Рис.15 Код для получения видео с YouTube в классе PatrubkiActivity



Рис.16 Код для доступа в интернет в манифесте проекта.



Рис.17. Код для добавления зависимости в Gradle.

По мимо поиска поломок, пользователь может найти сервисы официальных дилеров для проведения ремонта, если у него не получилось произвести ремонт самостоятельно. Для этого, на главном экране (Рис.9) ему необходимо выбрать пункт «Поиск сервисов», после чего откроется страница посика сервисов. Далее пользователь может выбрать сервис из списка предложенных, для этого ему необходимо нажать на название сервиса ,после чего произойдёт переход на страницу с подробной информацией о сервисе.

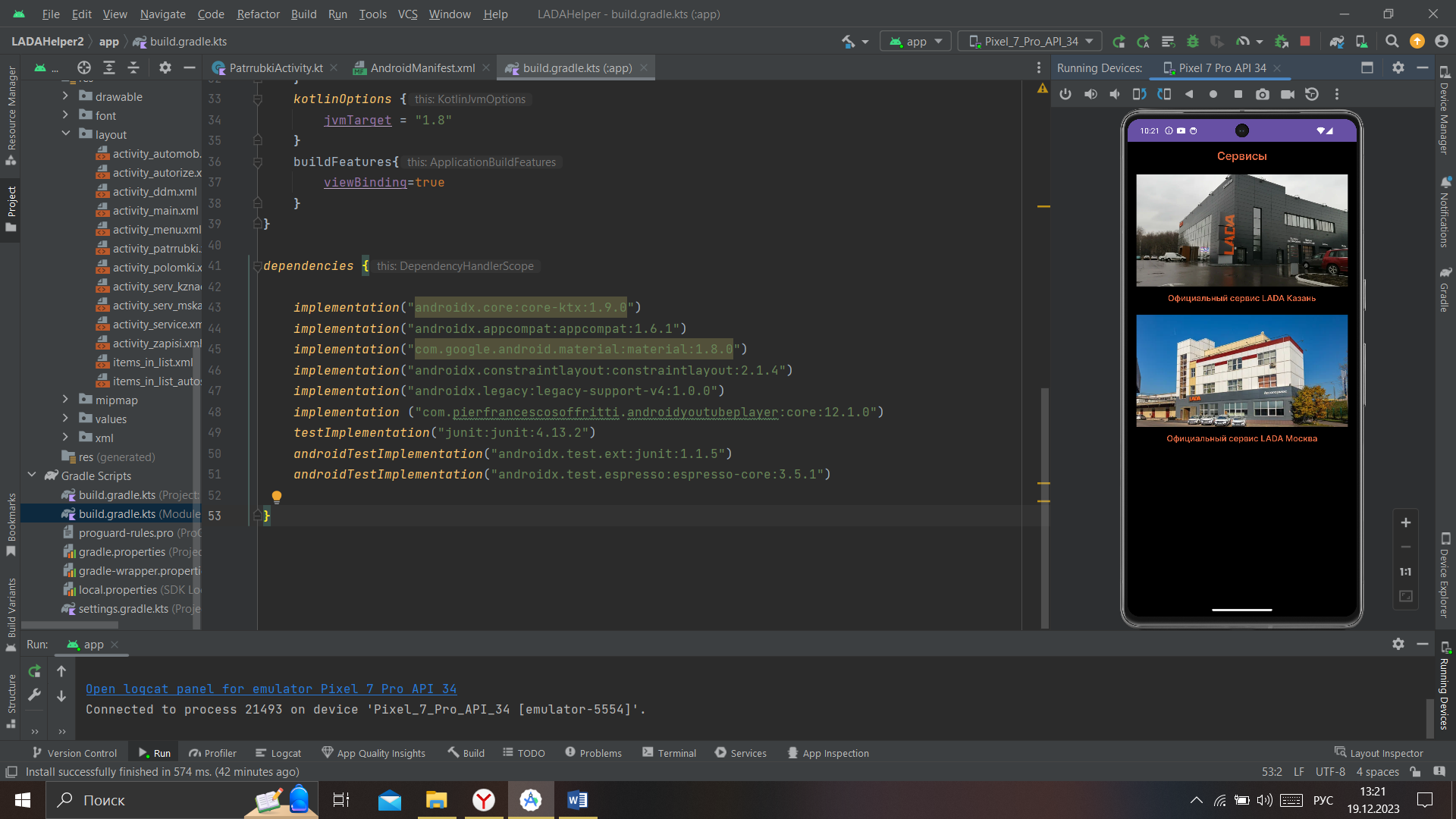


Рис.18 Страница выбора сервисов.

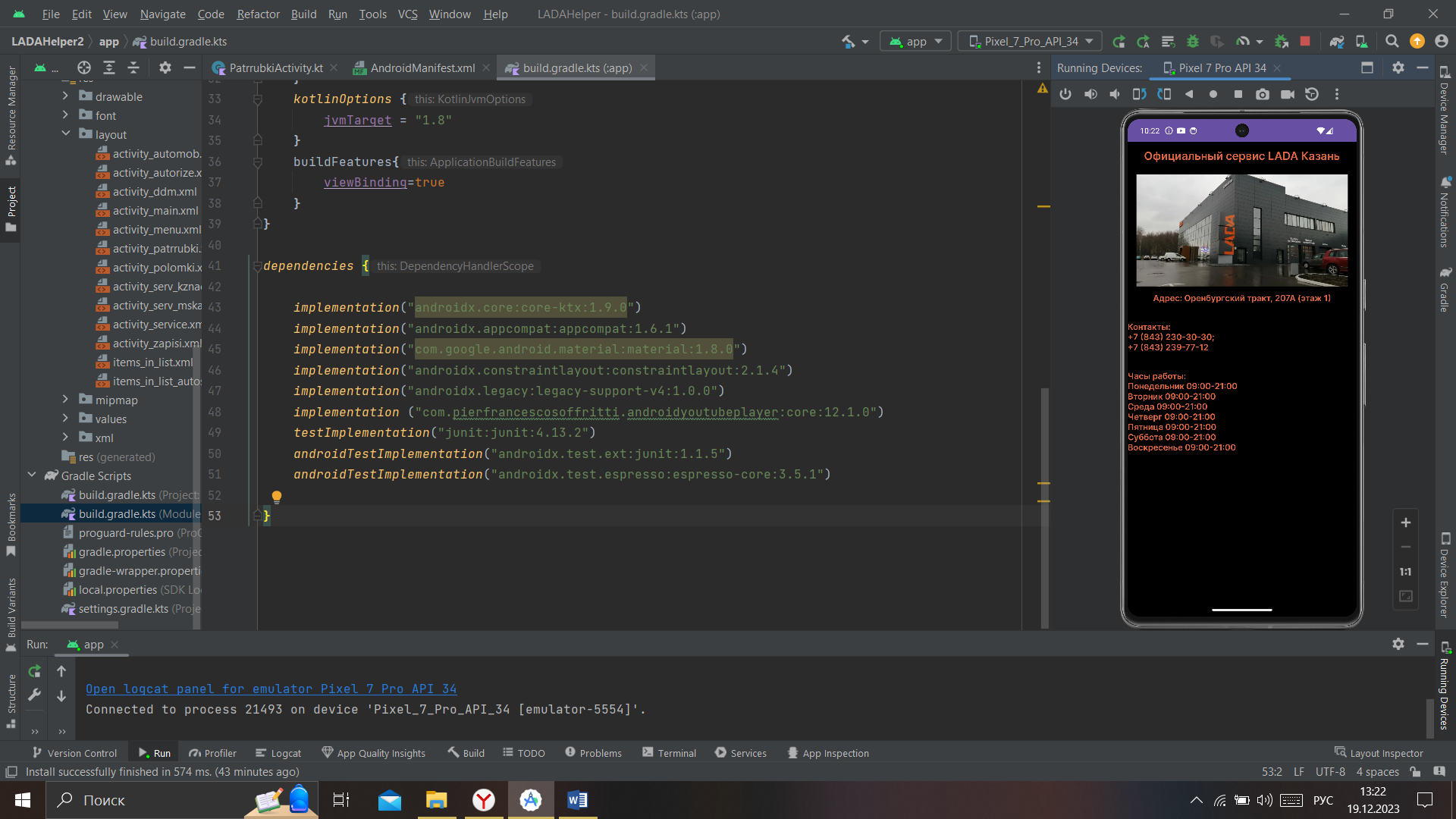


Рис.19 Страница с информацией о сервисе.

Следующим же функционалом является записи и заметки владельца о своем ремонте. Для перехода к записям пользователю необходимо на главном экране (Рисю9) выбрать пункт «Мои записи», после чего откроется страница записей, где пользователь может вести записи о проделанных работах.

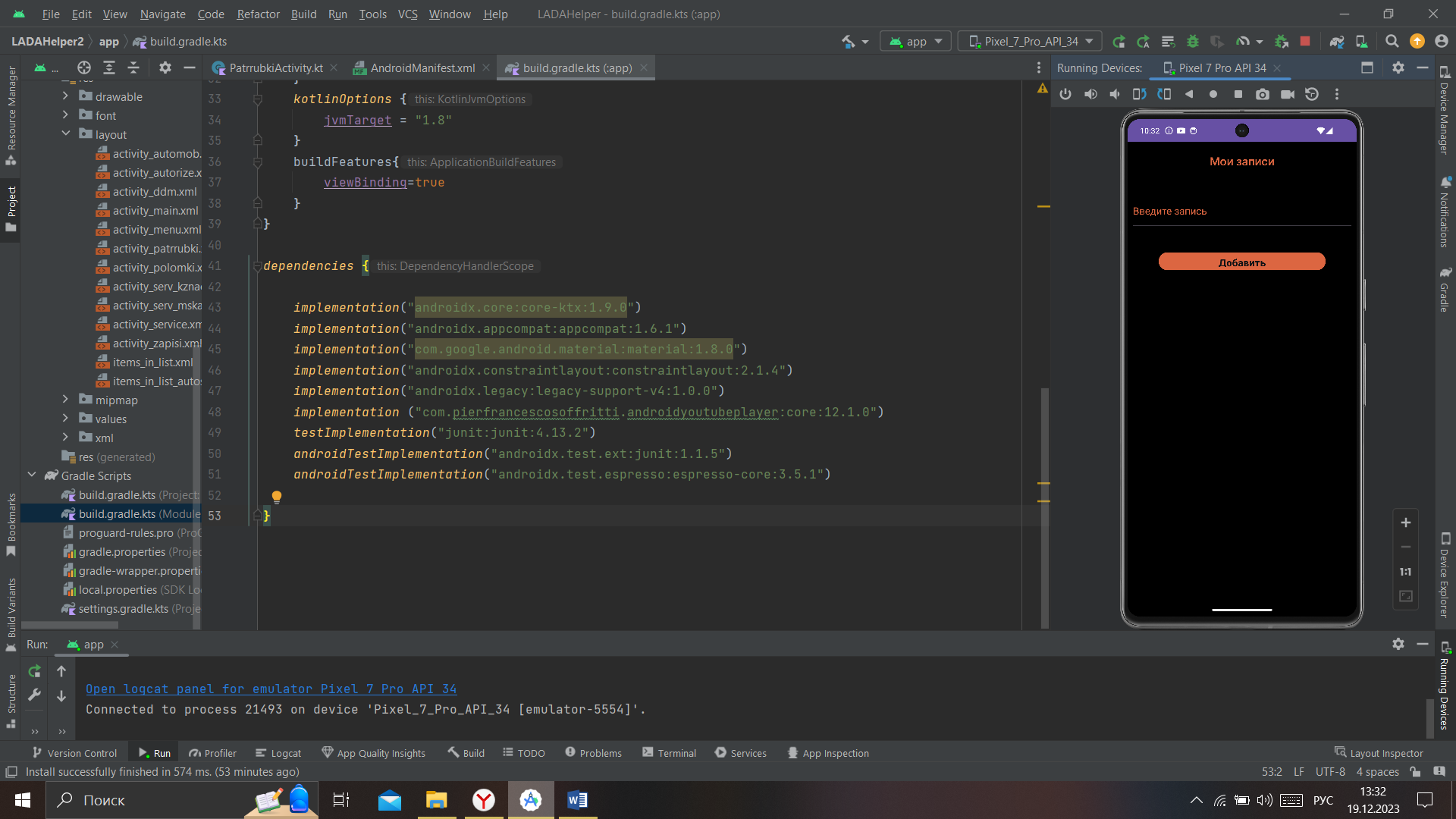


Рис.20 Страница с информацией о сервисе.

Записи пользователя сохраняются в базе данных, расположенной на его мобильном устройстве, аналогично записи логина и пароля, но уже в другой базе, где в таблице единственным элементом является сама запись.

Более того пользователю доступен такой функционал, как работа с картами, в приложении интегрированы карты от Yandex. Выбор именно этих карт обусловлен лучшей работой на территории России. При старте использования карт пользователю предоставляется выбор для определения место положения: точный, ориентировочный, и запрет на использование местоположения. После выбора определённого пункта, пользователю открываются доступ к самим картам, с помощью которых можно определить как лучше добраться до определенного места назначения, поскольку на картах отображены пробки в режиме реального времени. В случае если пользователь разрешил приложению определить место положение, приложение само определяет текущее место положение и приближает местность. К сожалению полный функционал в текущих условиях невозможно сделать, поскольку удалось получить только бесплатный ключ API от Yandex MapKit, полная же версия стоит довольно дорого, и в дальнейшем планируется также осуществить систему полной навигации. Визуал и код для работы всех систем находится уже в проекте.

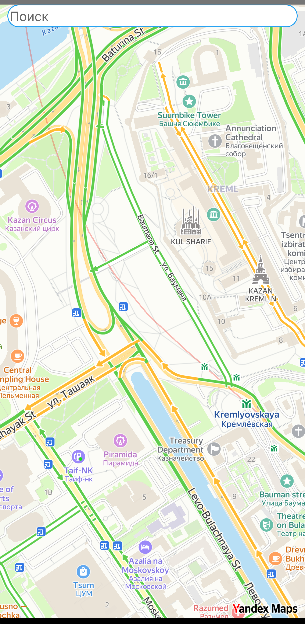
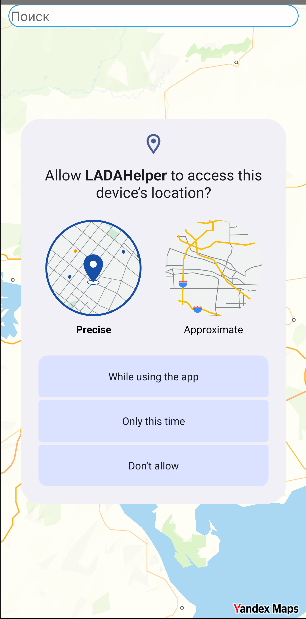
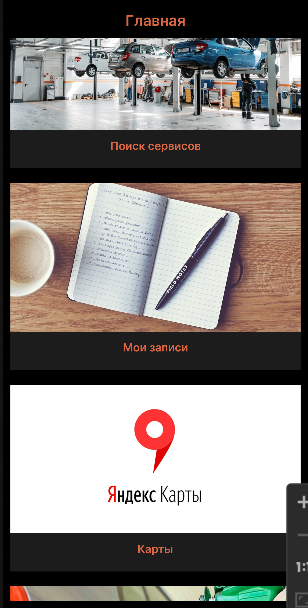


Рис.21. Использование карт в приложении



Рис.22. Инициализация карт

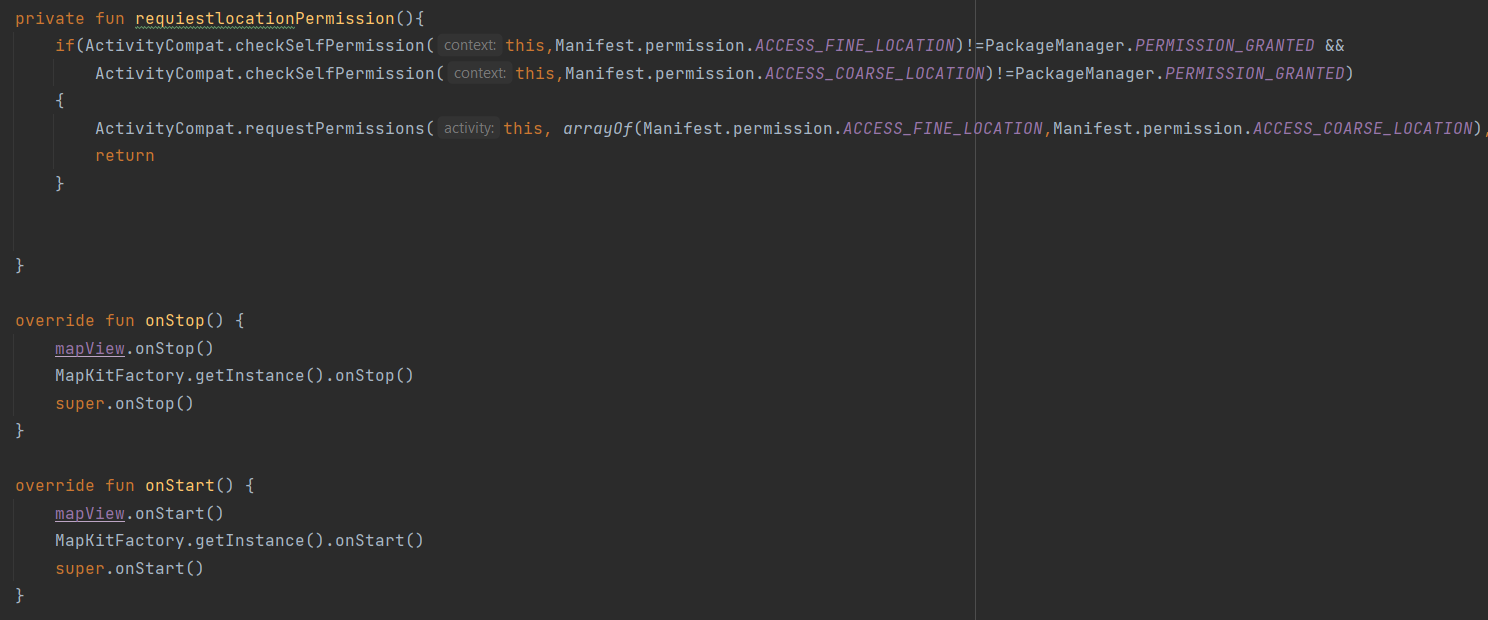


Рис.23. Функции для работы карт.

*implementation* ("com.yandex.android:maps.mobile:4.5.1-lite")

Рис.24. Зависимость для работы в Gradle.