Оглавление

[**Введение** 2](#_Toc122641711)

[**Первая глава** 3](#_Toc122641712)

[**Описание предметной области** 3](#_Toc122641713)

[**Глава 2. Результаты проектирования системы** 6](#_Toc122641714)

[**Диаграмма вариантов использования** 14](#_Toc122641715)

[**Диаграмма последовательностей** 19](#_Toc122641716)

[**Диаграмма развертывания** 20](#_Toc122641717)

[**Глава 3. Реализация системы** 22](#_Toc122641718)

[**Глава 4. Руководство пользователя и результат работы** 29](#_Toc122641719)

[**Заключение** 47](#_Toc122641720)

[**Список литературы** 48](#_Toc122641721)

[**Приложение** 49](#_Toc122641722)

# **Введение**

Темой данной курсовой работы является разработка приложения для сервиса технического обслуживания, в котором ведется учет клиентов (автомобилей), проводятся техническое обслуживание автомобиля, ремонт.

В современном мире каждая коммерческая организация стремится к увеличению КПД труда рабочих и увеличению прибыли. Данное приложение упростит работу по учёту клиентов, улучшит взаимодействие между различными отделами организации. И, что важнее всего, сможет исключить ненужные действия, что снизит число необходимых рабочих, но увеличит их награду. Именно поэтому выбранную задачу можно считать актуальной и необходимой.

В первой главе анализируется предметная область. Во второй главе обозреваются диаграммы вариантов использования, ер-диаграммы, диаграммы классов, диаграмма последовательностей и диаграмма состояний. В третьей главе описываются классы программы. В четвертой главе рассматривается инструкция пользователю.

# **Первая глава**

## **Описание предметной области**

В качестве предметной области выбрано СТО «Руки-крюки», который занимается обслуживанием автомобилей клиентов. СТО способно принимать клиентов (автомобили), хранить информацию о пройденных автомобилем ТО, отправлять автомобиль на техническое обслуживание или ремонт. К каждому ТО создаются необходимые услуги, которые содержат в себе запчасти. Программа «СТО «Руки-крюки». Работник» предназначена для добавления клиентов, проведения техосмотров и хранения записей о пройденных сервисах. Программа предоставляет возможность получения отчетов по ТО и используемых в них запчастей, по проведенным ТО за определенный период, с указанием запчастей и записей сервисов.

Данное программное решение было реализовано в версии для Веб-браузера для ПК.

## **Аналоги**

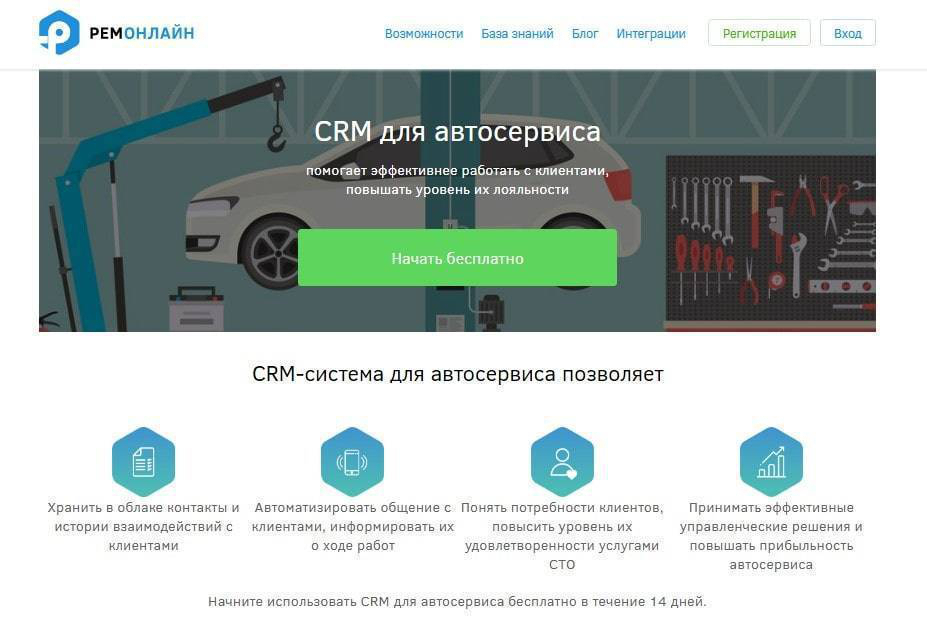
АвтоДилер

* Помогает автоматизировать работу автосервиса по расчету стоимости ремонта, планированию загрузки, общению с клиентами.
* Берет на себя работу с запчастями: подбор, заказ, складской учет, документы. Позволяет вести учёт материалов.
* Позволяет оценивать эффективность работы автосервиса, вести управленческий учет, увеличивать количество заездов клиентов, повышать средний чек и выручку. Есть наглядные отчёты и графики.



РемОнлайн

* Создание базы данных с возможностью группирования юридических, физических лиц, поставщиков и клиентов, удобный фильтр для быстрого поиска нужного клиента;
* Автоматизированное общение с клиентской базой, информирование их об этапах выполненных работ;
* Облако для хранения контактов и истории взаимодействий с клиентами и поставщиками;
* Быструю регистрацию новых клиентов и обработку заказов благодаря готовым шаблонам карточек;
* Создание предварительных записей клиентов на ТО и ремонт, фиксирование продолжительности выполнения работ, быстрый поиск свободного или необходимого клиенту мастера;
* Интеграция с сервисом IP-телефонии и контроль качества обслуживания клиентов по телефону;
* Оповещение сотрудников и клиентов в автоматическом режиме при помощи SMS или email-сообщений.



## **Техническое задание на разработку программного продукта «СТО «Руки-Крюки». Поставщик»**

Программа «СТО «Руки-крюки». Работник» предназначена для ведения учета обслуживания машин. Работники указывают доступные ТО, машины, которые их проходили, и записи сервисов машин. Программа предоставляет возможность получения отчётов по ТО.

**Задачи:**

**-**формирование записи (CRUD) по «ТО» с выбором из списка «работ» нескольких записей к ней;

-получение списка по «запчастям» на основе выбранных записей «ТО» в формате doc/xls;

-формирование записей (CRUD) «машин»;

-привязывание записей «машин» к записям «ТО»;

-формирование записей (CRUD) «сервисов» с привязкой записи «ТО»;

-получение отчёта за период по записям из «ТО» с расшифровкой по записям «запчастей», относящихся к записям «машин», относящихся к записям «ТО» в формате pdf на почту и вывод отчёта на форму.

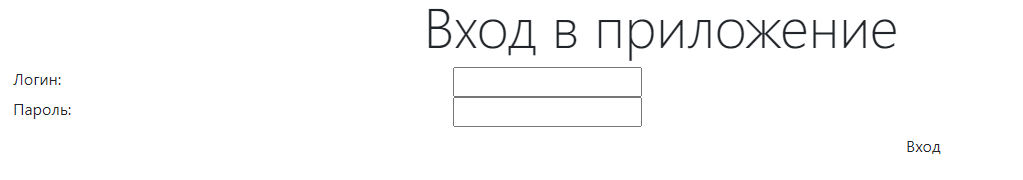
**Основной функционал программы:**

* Регистрация. Для регистрации пользователь должен заполнить следующие данные:

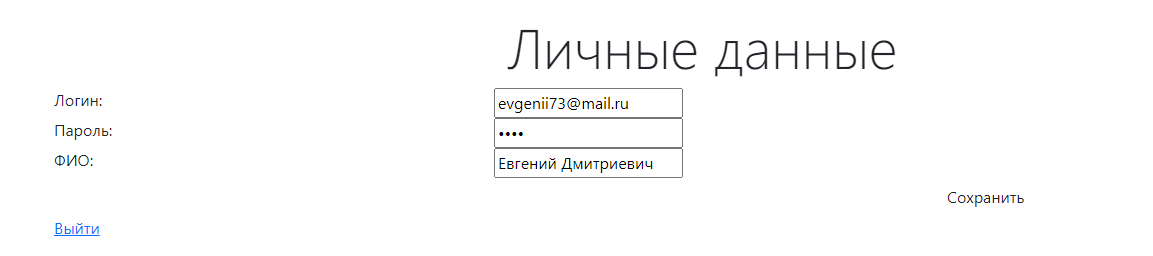
Логин, пароль, повтор пароля (пароль – не менее 8 символов, совпадение введённых паролей)



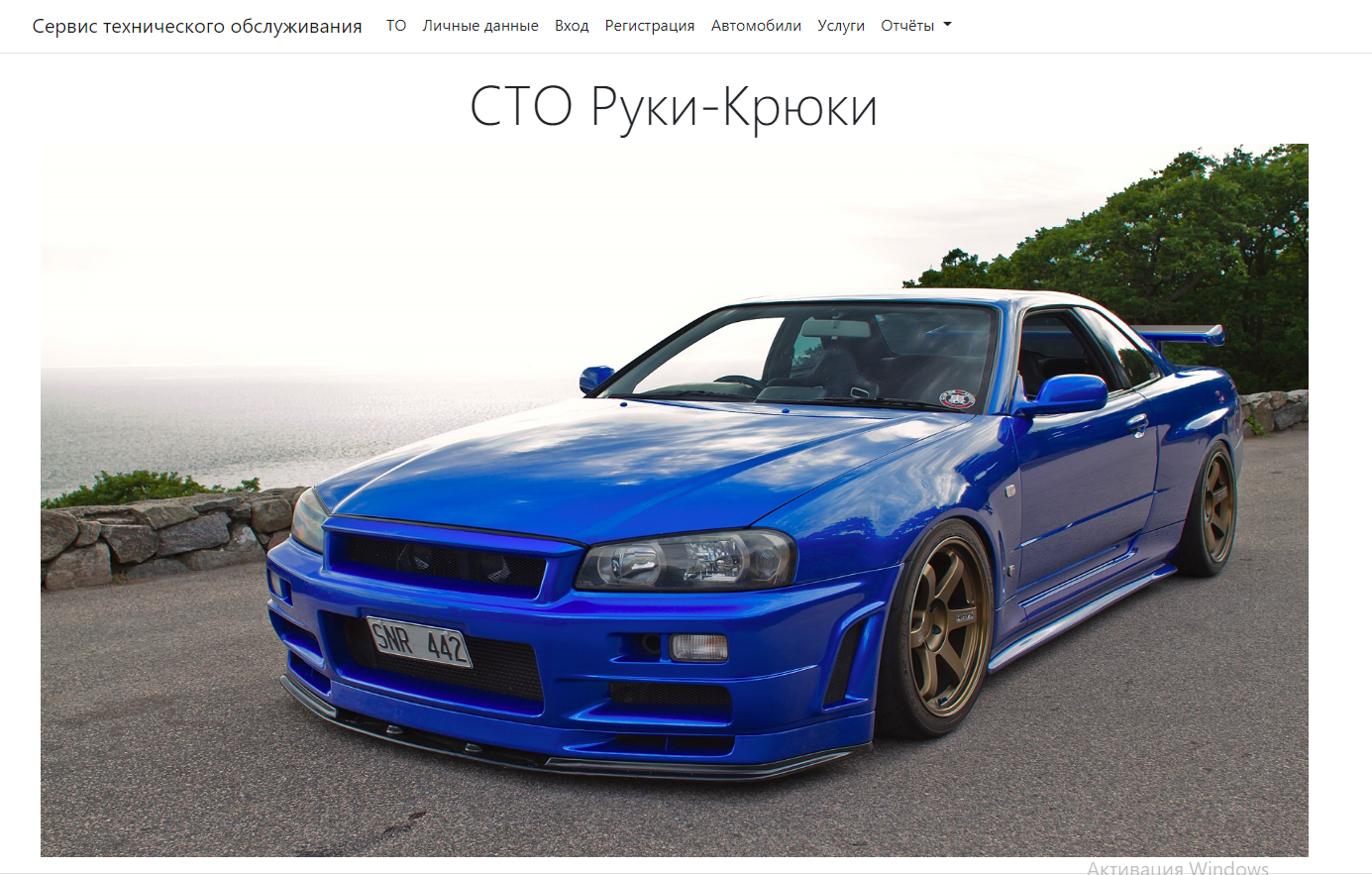
* Авторизация. Пользователь вводит данные. Если такой пользователь есть в системе, то происходит переход к основной форме. Если такого пользователя нет в системе, выводится сообщение об этом с просьбой ввести заново данные.



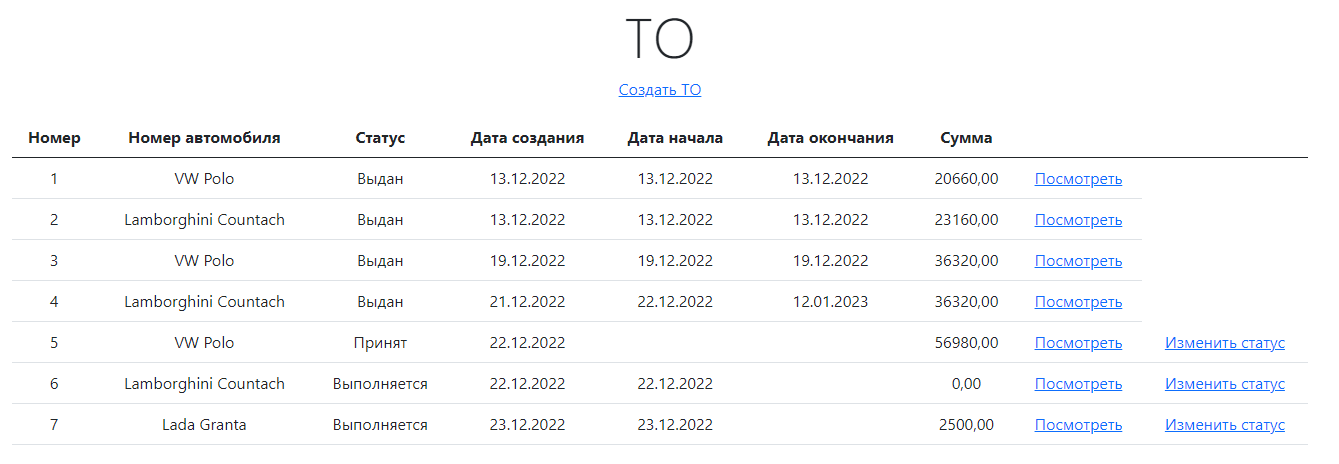
* Личные данные. Выводятся данные пользователя о его аккаунте (пароль, логин, ФИО). Есть возможность выйти из аккаунта или обновить данные.



* Основная форма. Состоит из пунктов меню и логотипа. Через пункты меню можно перейти на формы работы с ТО, машинами (в том числе и привязка машин), записями сервисов, на форму получения списка запчастей и на форму получения отчета.

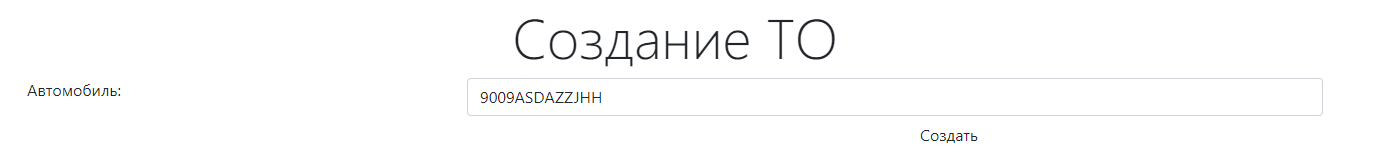


* Список всех ТО. Имеется форма, на которой расположены все существующие ТО, относящиеся к конкретному работнику. Можно создать новое ТО, изменить/посмотреть существующее, изменить статус ТО.

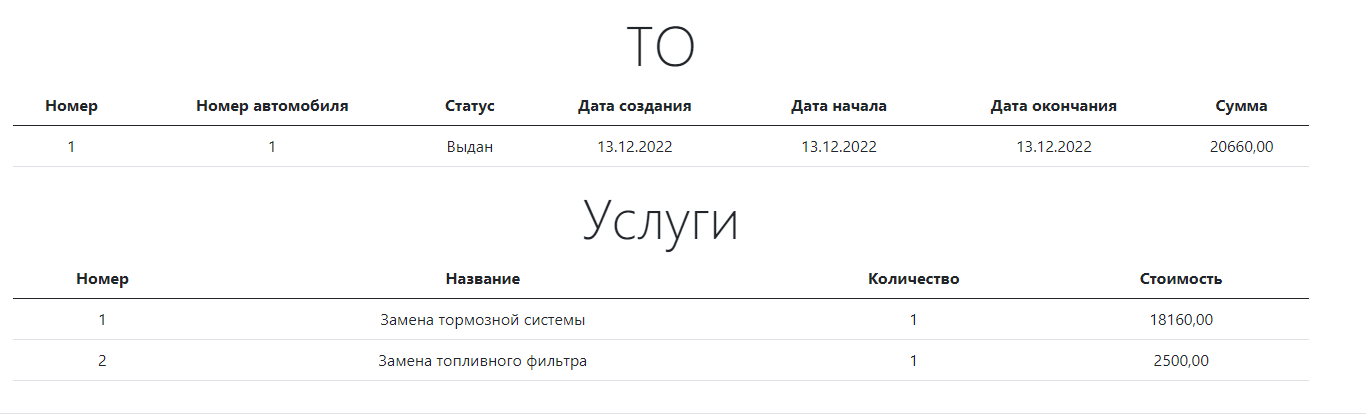


* Создание ТО (CRUD). Имеется форма. Пользователь может создать запись ТО, заполнив следующее поле:

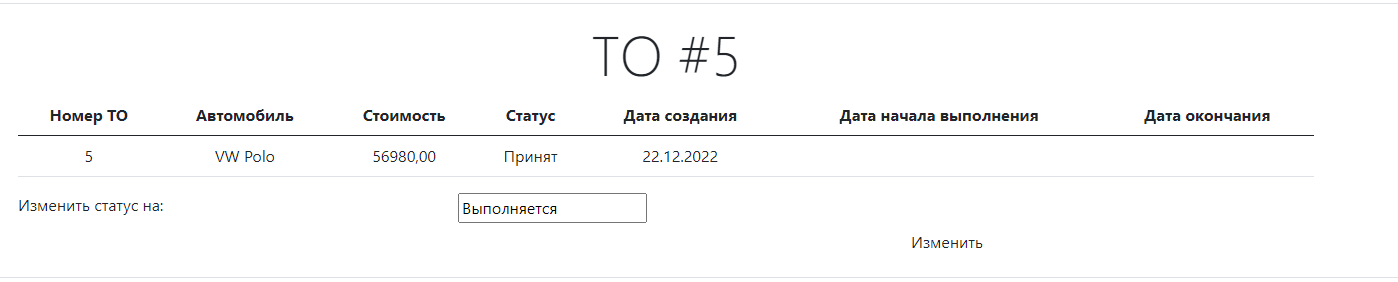
1. VIN-номер авто;



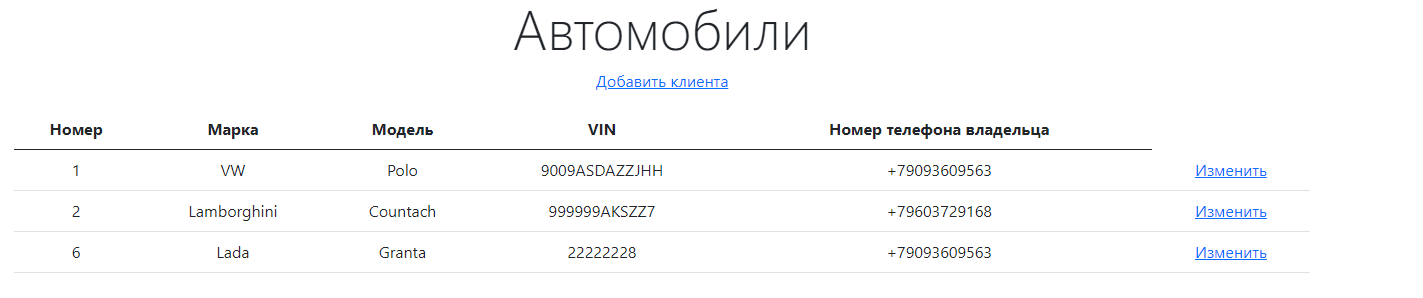
* Просмотр ТО. На форму выводятся данные о ТО и содержащиеся в нем услуги.



* Изменение статуса ТО. На форму выводятся данные о ТО. Есть возможность изменить статус ТО.

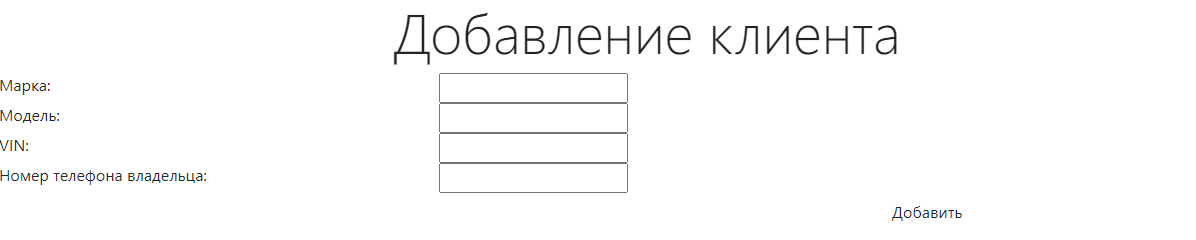


* Список всех автомобилей. На форму выводятся все автомобили, являющиеся клиентами СТО. Есть возможность добавить новый автомобиль, изменить запись.

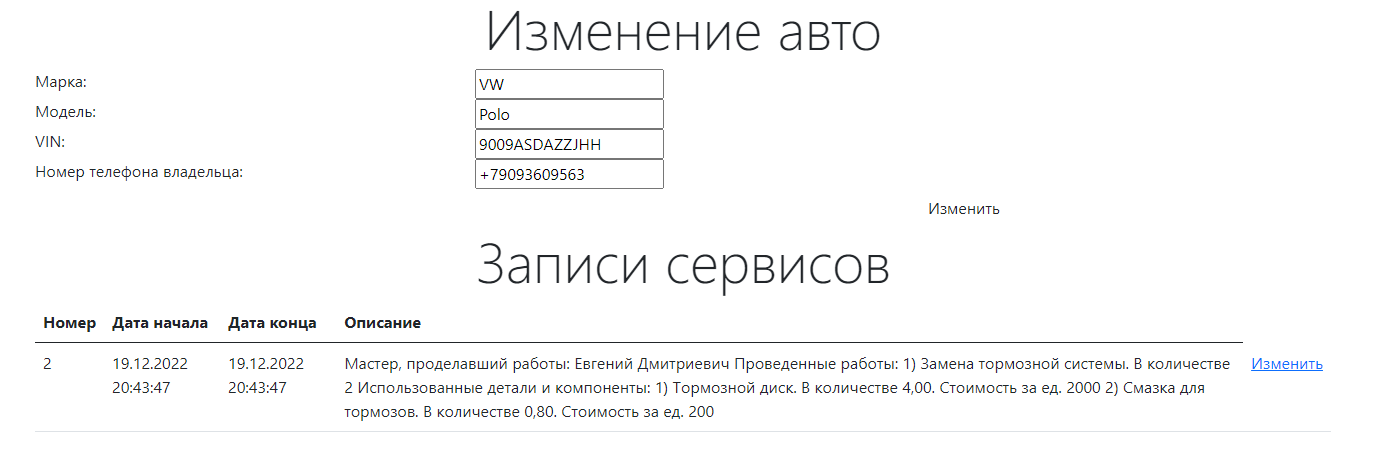


* Добавление автомобиля (CRUD). Выводится форма, в которой необходимо заполнить следующие поля:

1. Марка
2. Модель
3. VIN
4. Номер телефона владельца



* Изменение данных автомобиля. На форму выводятся данные об авто и связанные с ним записи сервисов. Есть возможность изменить данные автомобиля или изменить записи его сервисов.



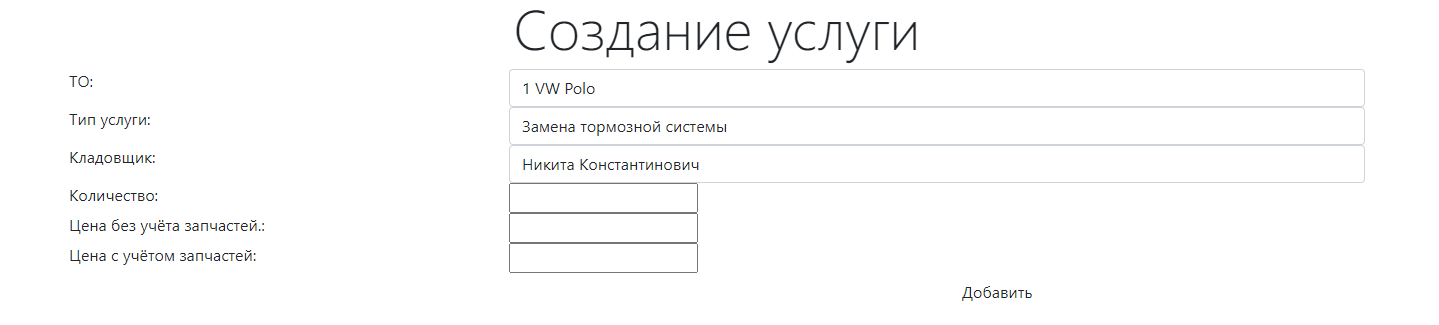
* Список всех услуг. На форму выводятся все услуги, связанные с ТО, которые проводит работник. Здесь можно следить за их прогрессом выполнения. А также просмотреть запчасти, используемые в той или иной услуге. Также есть возможность создать услугу.

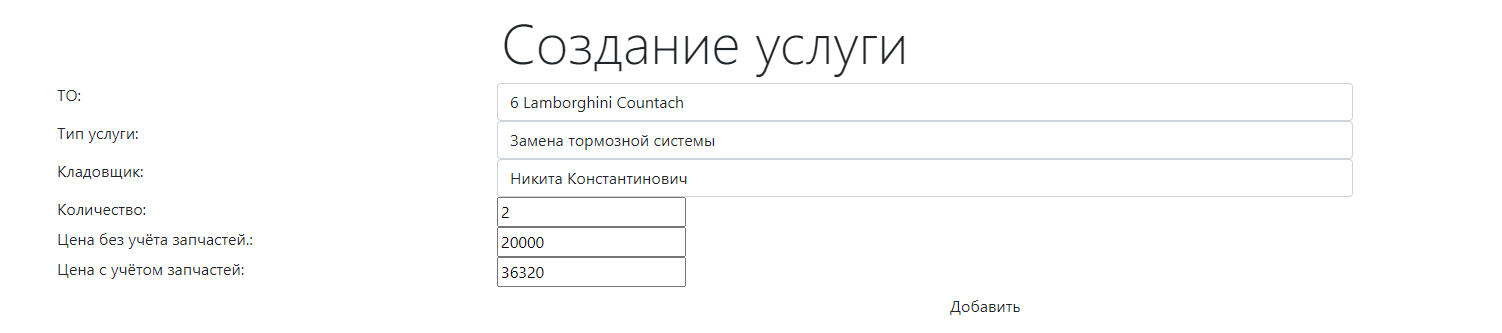


* Создание услуги (CRUD). Выводится форма, на которой необходимо заполнить следующие поля:

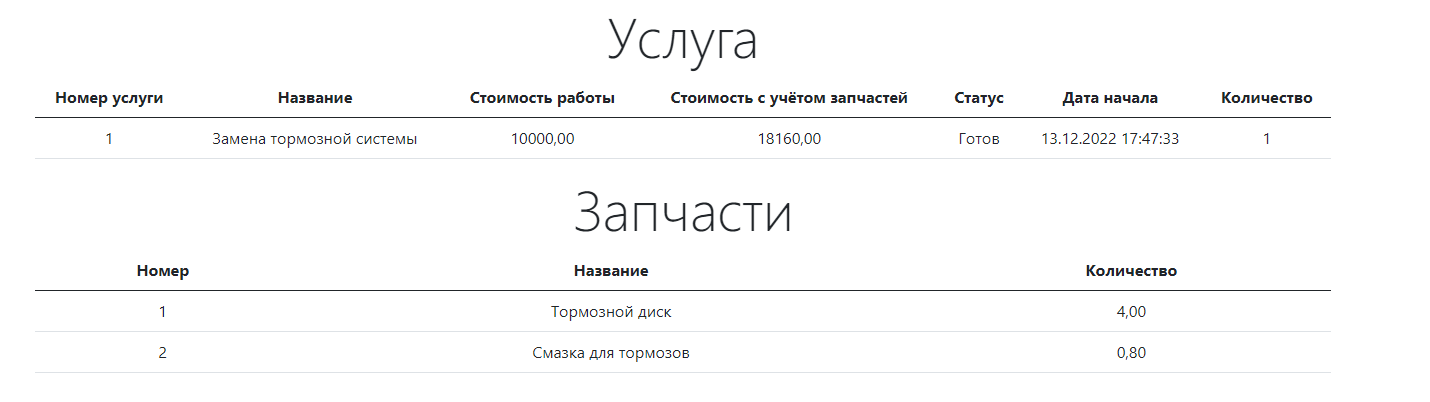
1. ТО
2. Тип услуги
3. Кладовщик
4. Количество

Поля цена без учёта запчастей и цена с учётом запчастей заполнятся автоматически.

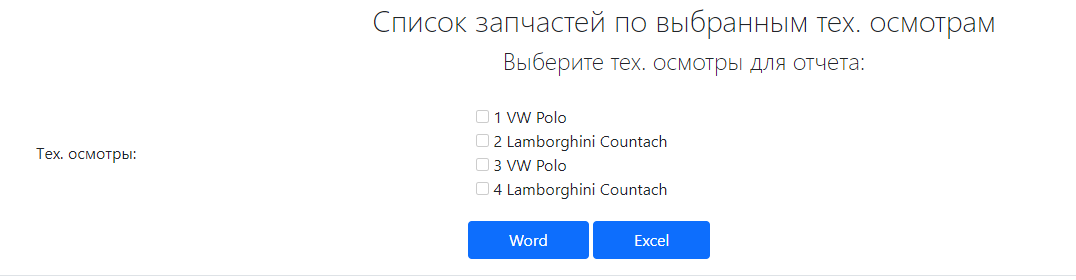




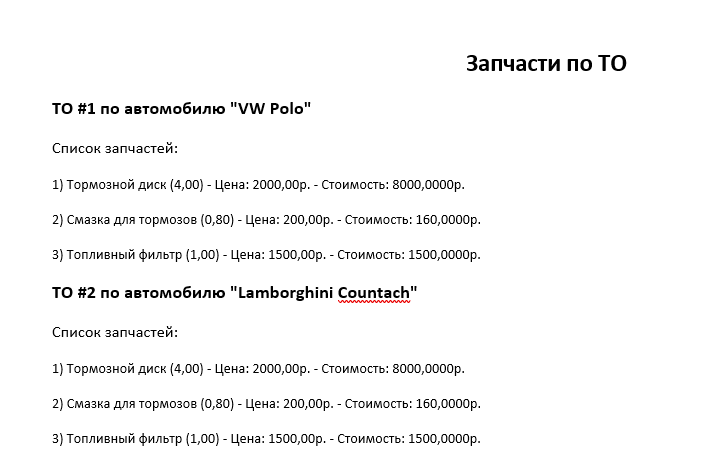
* Просмотр услуги. На форму выводятся данные об услуги и используемые в ней запчасти.



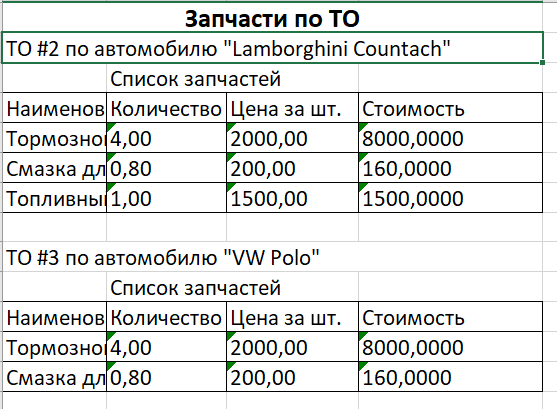
* Получение списка. Пользователь может получить список запчастей по выбранным ТО. Пользователь отмечает интересующие его ТО, указывает формат файла (docx или xlsx) и получает интересующий его список.



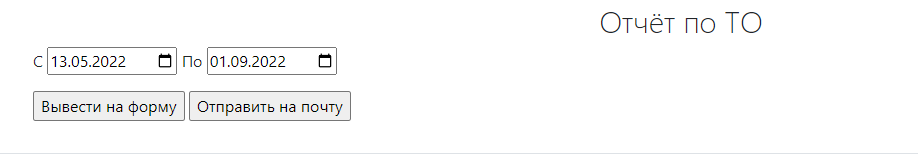
При выводе в docx-файл формат документа такой:

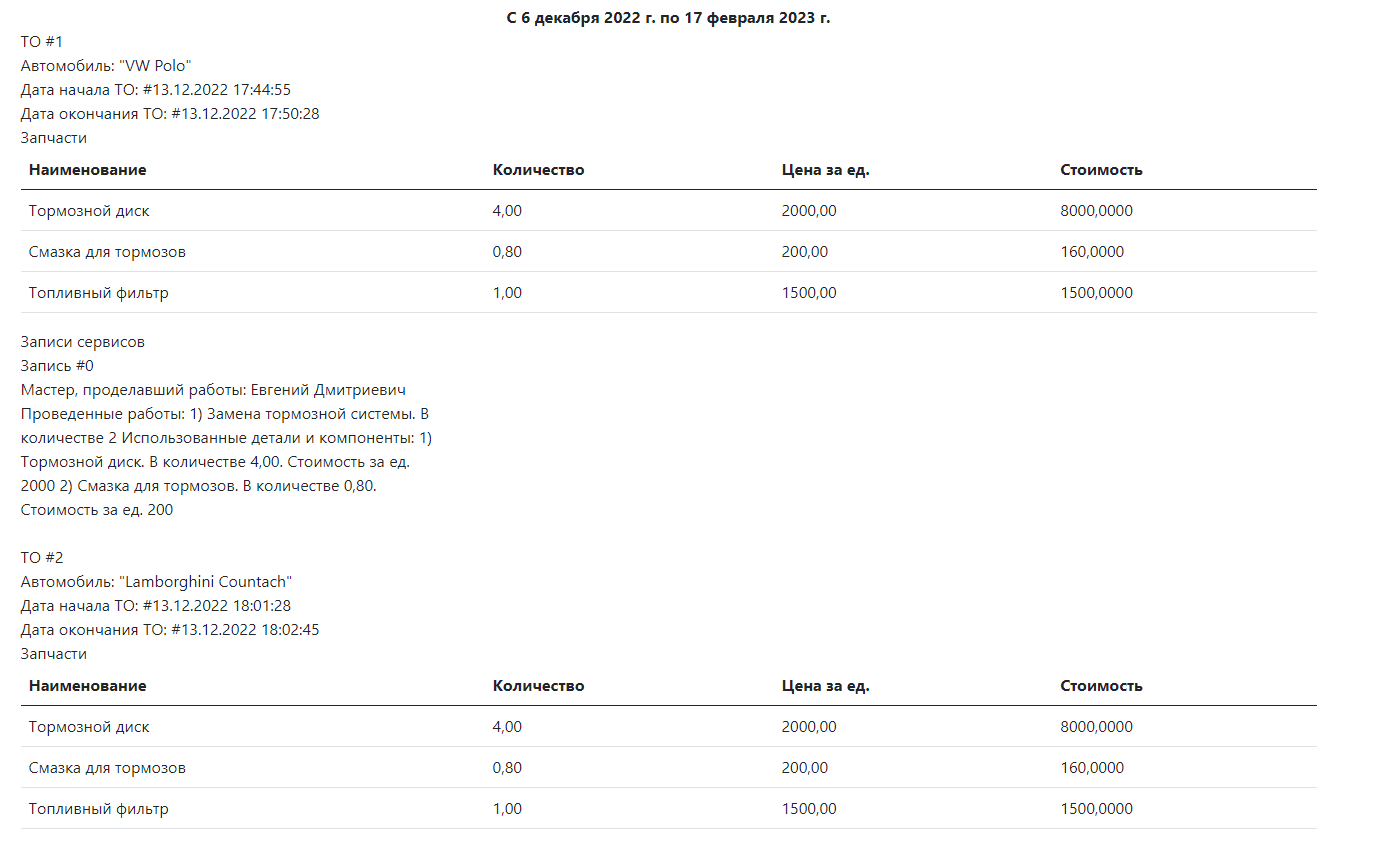
****

При выводе в xlsx-файл формат документа такой:



* Пользователь может получить сведения по полученным им запчастям и полученным записям сервисов за период по различным ТО. Для этого он должен указать интересующий его период, выбрать вариант получения сведений (отправка на почту или вывод на форму) и получить отчёт.





В качестве интерфейса пользователя будет выступать web-приложение, разработанное на ASP .NET Core.

# **Глава 2. Результаты проектирования системы**

## **Диаграмма вариантов использования**

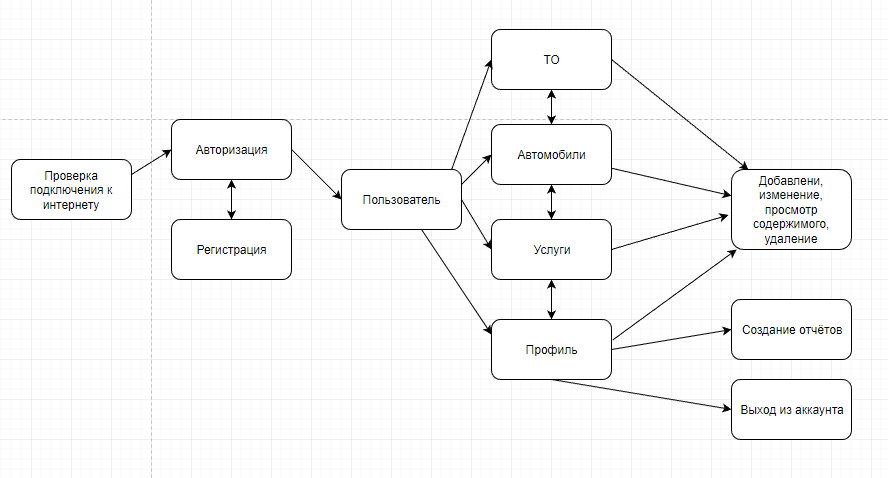


Рисунок 1. Диаграмма вариантов использования

Пользователем программы является работник сервиса. При входе в приложение необходимо зарегистрироваться или авторизоваться. В главном окне можно управлять текущими и завершенными ТО. Можно переключаться между экранами с помощью нижнего меню. При вводе неверных данных пользователю выведется ошибка. Также есть возможность получить отчеты. Также есть разграничение по ролям, если работник заходит под ролью “Старший работник”, то ему становятся доступны все текущие ТО, а не только проводимые им.

**ER-диаграмма**

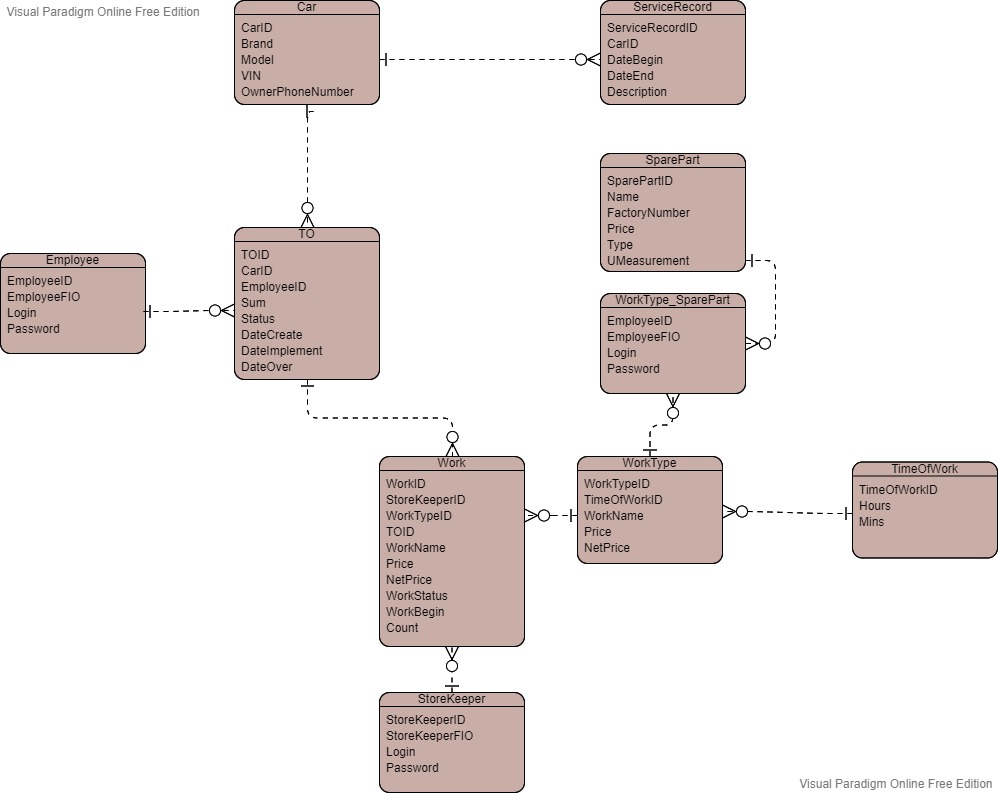


Рисунок 2. ER-диаграмма

В данной диаграмме представлена схема базы данных для всего проекта “СТО”Руки-крюки.””

Описание таблиц:

«Работник» (Employee) – сущность, содержащая информацию о работнике СТО.

Атрибуты:

* EmployeeID – идентификационный номер. Уникальный для каждой записи.
* Login – логин.
* Password – пароль.
* EmployeeFIO – фио.

«ТО» (TO) – сущность, содержащая информацию о ТО.

Атрибуты:

* TOID – идентификационный номер. Уникальный для каждой записи.
* CarID - идентификационный номер связанного авто.
* EmployeeID - идентификационный номер работника.
* Sum – стоимость.
* Status – статус.
* DateCreate – дата создания
* DateImplement – дата начала работ
* DateOver – дата окончания

«Автомобиль» (Car) - сущность, содержащая информацию об автомобиле

Атрибуты:

* CarID – идентификационный номер. Уникальный для каждой записи.
* Brand – марка авто.
* Model – модель авто.
* VIN – VIN-номер авто.
* OwnerPhoneNumber – номер телефона владельца.

«Запись сервиса» (ServiceRecord) - сущность, содержащая информацию о записи сервиса.

Атрибуты:

* ServiceRecordID – идентификационный номер. Уникальный для каждой записи.
* CarID – идентификационный номер автомобиля, связанного с записью.
* DateBegin – дата начала проведения ТО, связанного с записью.
* DateEnd – дата окончания проведения ТО, связанного с записью.
* Description – описание проведенного ТО.

«Услуга/Работа» (Work) - сущность, содержащая информацию об услуге.

Атрибуты:

* WorkID – идентификационный номер. Уникальный для каждой записи.
* StoreKeeperID – идентификационный номер кладовщика, связанного с записью.
* WorkTypeID – идентификационный номер типа услуги, связанного с услугой.
* TOID– идентификационный номер ТО, связанного с услугой.
* WorkName – название.
* Price – стоимость проведения услуги.
* NetPrice – стоимость услуги + запчастей.
* WorkStatus – статус услуги.
* WorkBegin – дата начала проведения услуги.
* Count – количество.

«Тип услуги» (WorkType) - сущность, содержащая информацию о типе услуги.

Атрибуты:

* WorkTypeID – идентификационный номер. Уникальный для каждой записи.
* TimeOfWorkID – идентификационный номер времени работы, связанного с типом услуги.
* WorkName – название
* Price – стоимость услуги.
* NetPrice – стоимость услуги + запчастей.

«Время работы» (TimeOfWork) - сущность, содержащая информацию о времени работы.

Атрибуты:

* TimeOfWorkID – идентификационный номер. Уникальный для каждой записи.
* Hours – часы.
* Mins – минуты.

«ТипРаботы\_Запчасть» (WorkType\_SparePart) - сущность, содержащая информацию о связанности типа услуги и запчастей.

Атрибуты:

* ID - идентификационный номер. Уникальный для каждой записи.
* WorkTypeID – идентификационный номер типа услуги.
* SparePartID – идентификационный номер запчасти.
* Count – количество запчастей.

«Запчасть» (SparePart) - сущность, содержащая информацию о запчасти.

Атрибуты:

* SparePartID - идентификационный номер. Уникальный для каждой записи.
* Name – название.
* FactoryNumber – заводской номер.
* Price – цена.
* Type – тип.
* UMeasurement – ед. измерения.

## **Диаграмма последовательностей**

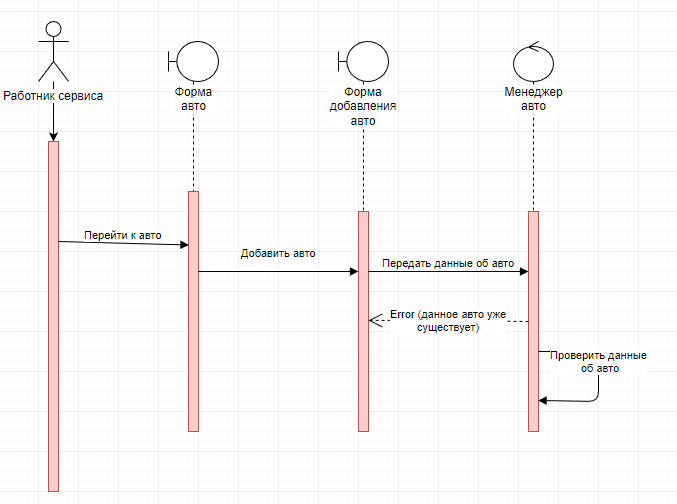


Рисунок 3. Диаграмма последовательности

На данной диаграмме предоставлена последовательность работы приложения на примере добавления автомобиля (клиента). В начале происходит ввод данных об авто. Далее данные передаются в БД и проверяются. Если нет ошибок, то создается новая запись. После проверки данных происходит связывание, и работник получает сообщение о том, что связывание прошло успешно.

## **Диаграмма развертывания**

Программа имеет следующую структуру:

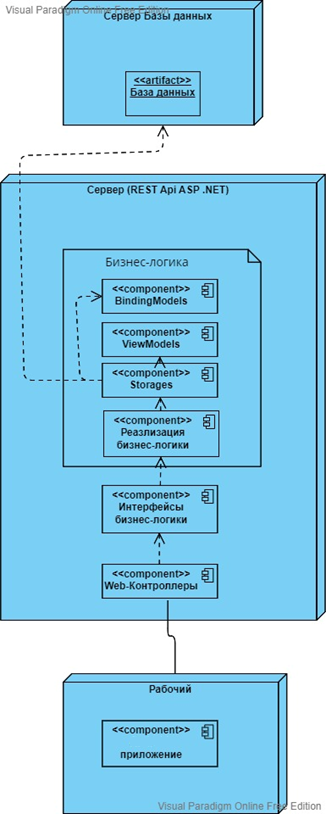


Рисунок 4. Диаграмма развертывания

Сервер БД и сервер REST Api, который обращается к бизнес-логике, которая обращается к логике хранения сущностей и взаимодействует с моделями (создаёт их, изменяет, удаляет). На диаграмме отображены:

* Классы бизнес-логики – описание логики работы, принципы CRUD
* CRUD – описание функций (запросов) для работы с БД (создание, чтение, изменение, удаление)
* BindingModels – классы для создания, изменения, удаления сущностей, которые сервер REST Api получает от клиентов.
* ViewModels – классы для представления сущностей пользователю. REST Api отправляет их клиенту.
* Интерфейсы для хранилищ (описание методов для получения списков, элементов, добавление/изменение записей). Взаимодействует непосредственно с базой данных.
* EntityFrameworkCore для работы с БД.

# **Глава 3. Реализация системы**

Системы была реализована с помощью паттерна MVC (Модель-Представление-Контроллер). MVC – схема разделения данных приложения и управляющий логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Controller – пользователь (приложение) обращается к контроллерам REST Api для изменения, удаления и добавления данных. Контроллер в свою очередь обращается к Model, которая содержит в себе всю логику и манипулирует базой данных (ASP.NET Core).

Model – бизнес-логика и логика управления БД. Манипулирует сущностями (Microsoft SQL Express, Microsoft.EntityFrameworkCore).

View – логика отправки данных пользователю. REST Api отправляет пользователю удобопредставленные данные, с которыми затем он может взаимодействовать и отправлять запрос на изменение или создание обратно на сервер. На устройстве пользователя всё предстаёт в человекопонятном виде.

Программа имеет следующую структуру:

1. Service – папка, содержащая интерфейс и класс, для взаимодействия пользовательского приложения с RestApi.



Рисунок 5. Service

1. service
   1. STOService – класс для подключения к RestApi на сервере
   2. STOApi – интерфейс, описывающий взаимодействие с RestApi
2. Models

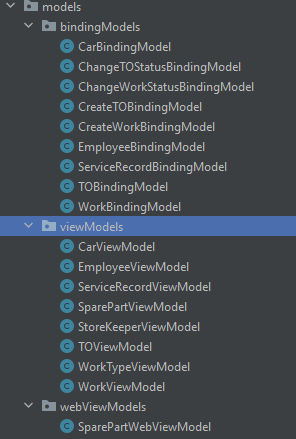


Рисунок 6. Модели

* 1. bindingModels – папка, содержащая binding модели.
     1. CarBindingModel – класс, описывающий авто. Содержит поля Id, Brand, Model, VIN, OwnerPhoneNumber, Records и геттеры, сеттеры для них.
     2. ChangeTOStatusBindingModel – класс, описывающий модель изменения статуса ТО. Содержит поле TOId и геттеры, сеттеры для него.
     3. CreateTOBindingModel – класс, описывающий модель для создания ТО. Содержит поля CarId, EmployeeId, Sum и геттеры, сеттеры для них.
     4. CreateWorkBindingModel – класс, описывающий модель для создания услуги. Содержит поля StoreKeeperId, WorkTypeId, TOId, Count, Price, NetPrice и геттеры, сеттеры для них.
     5. EmployeeBindingModel – класс, описывающий работника. Содержит поля Id, FIO, Login, Password и геттеры, сеттеры для них.
     6. ServiceRecordBindingModel – класс, описывающий запись сервиса. Содержит поля Id, CarId, DateBegin, DateEnd, Description и геттеры, сеттеры для них.
     7. TOBindingModel – класс, описывающий ТО. Содержит поля Id, CarId, EmployeeId, Sum, Status, DateCreate, DateImplement, DateOver и геттеры, сеттеры для них.
     8. WorkBindingModel – класс, описывающий услугу. Содержит поля Id, StoreKeeperId, WorkTypeId, TOId, WorkName, Price, NetPrice, WorkStatus, WorkBegin, Count и геттеры, сеттеры для них.
  2. viewModels – папка, содержащая view модели.
     1. CarViewModel – класс, описывающий авто. Содержит поля Id, Brand, Model, VIN, OwnerPhoneNumber, Records и геттеры, сеттеры для них.
     2. EmployeeViewModel – класс, описывающий работника. Содержит поля Id, FIO, Login, Password и геттеры, сеттеры для них.
     3. ServiceRecordViewModel – класс, описывающий запись сервиса. Содержит поля Id, CarId, CarBrandAndName, DateBegin, DateEnd, Description и геттеры, сеттеры для них.
     4. SparePartViewModel – класс, описывающий запчасть. Содержит поля Id, Name, FactoryNumber, Price, Type, UMeasurement и геттеры, сеттеры для них.
     5. StoreKeeperViewModel – класс, описывающий кладовщика. Содержит поля Id, FIO, Login, Password и геттеры, сеттеры для них.
     6. TOViewModel – класс, описывающий ТО. Содержит поля Id, CarId, CarName, EmployeeId, EmployeeName, TOAndCarName, Sum, Status, DateCreate, DateImplement, DateOver, Works и геттеры, сеттеры для них.
     7. WorkTypeViewModel – класс, описывающий тип услуги. Содержит поля Id, TimeOfWorkId, ExecutionTime, WorkName, Price, NetPrice, WorkSpareParts и геттеры, сеттеры для них.
     8. WorkViewModel – класс, описывающий услугу. Содержит поля Id, StoreKeeperId, StoreKeeperFIO, WorkTypeId, TOId, WorkName, Price, NetPrice, WorkStatus, WorkBegin, Count и геттеры, сеттеры для них.
  3. webViewModels – папка, содержащая web view модели.
     1. SparePartWebViewModel – класс, описывающий деталь. Содержит поля Count, Name, FactoryNumber, Price, Type, UMeasurement и геттеры, сеттеры для них.
  4. enums – папка, содержащая необходимые перечисления.



Рисунок 7. Перечисления

* + 1. TOStatus – перечисления для статусов ТО.
    2. WorkStatus – перечисления для статусов Услуги.
  1. Adapters – папка, содержащая классы для адаптеров для Recyclerview.

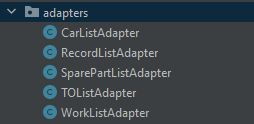


Рисунок 8. Адаптеры

* + 1. CarListAdapter – класс-адаптер для элемента recyclerview для отображения авто.
    2. RecordListAdapter – класс-адаптер для элемента recyclerview для отображения записи сервиса.
    3. SparePartListAdapter – класс-адаптер для элемента recyclerview для отображения запчасти.
    4. TOListAdapter – класс-адаптер для элемента recyclerview для отображения ТО.
    5. WorkListAdapter – класс-адаптер для элемента recyclerview для отображения услуги.
  1. Spinners – папка, содержащая классы для адаптеров для Spinner.

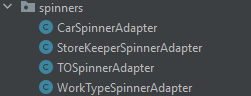


Рисунок 9. Адаптеры для Spinner

* + 1. CarSpinnerAdapter – класс-адаптер для элемента spinner для авто.
    2. StoreKeeperSpinnerAdapter – класс-адаптер для элемента spinner для кладовщика.
    3. TOSpinnerAdapter – класс-адаптер для элемента spinner для ТО.
    4. WorkTypeSpinnerAdapter – класс-адаптер для элемента spinner для типа услуги.
  1. Ui – папка, хранящая визуальные элементы (фрагменты).
     1. cars – папка, хранящая фрагменты для авто.

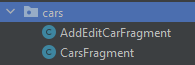


Рисунок 10. Фрагменты авто

* + - 1. AddEditCarFragment – фрагмент для создания/редактирования авто.
      2. CarsFragment – фрагмент, показывающий авто.
    1. logregfragments – папка, содержащая фрагменты для регистрации/авторизации



Рисунок 11. Фрагменты авторизации, регистрации

* + - 1. LogInFragment – фрагмент для авторизации.
      2. RegisterFragment – фрагмент для регистрации.
    1. records – папка, содержащая фрагменты для записи сервисов.



Рисунок 12. Фрагменты записи сервисов

* + - 1. EditFragment – фрагмент для редактирования записи сервисов.
    1. to – папка, содержащая фрагменты для ТО.

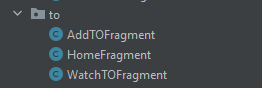


Рисунок 13. Фрагменты ТО

* + - 1. AddTOFragment – фрагмент для создания ТО.
      2. HomeFragment – фрагмент для просмотра всех ТО.
      3. WatchFragment – фрагмент для взаимодействия с ТО.
    1. works – папка, содержащая фрагменты для услуг.

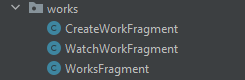


Рисунок 14. Фрагменты услуг

* + - 1. CreateWorkFragment – фрагмент для создания услуги.
      2. WatchWorkFragment – фрагмент для просмотра запчастей услуги.
      3. WorksFragment – фрагмент для просмотра всех работ.
    1. ProfileFragment – фрагмент для просмотра данных о профиле и для создания отчётов.



Рисунок 15. Фрагменты профиля

* 1. App – класс-application. Содержит данные об авторизованном пользователе и предоставляет ему взаимодействие с приложением. А также содержит логику для взаимодействия с сервисом.
  2. LogInRegisterActivity – активити, содержащая фрагменты с авторизацией и регистрацией.
  3. MainActivity – активити, содержащая все остальные фрагменты.

# **Глава 4. Руководство пользователя и результат работы**

Заставочный экран



Рисунок 16. Заставочный экран

Регистрация

Для регистрации пользователь должен заполнить:

1. Логин
2. ФИО
3. Пароль
4. Повтор пароля

Затем эти данные отправляются на сервер. При корректных данных приходит положительный ответ, что пользователь зарегистрировался.



Рисунок 17. Фрагмент регистрации

Авторизация

Для авторизации пользователь должен заполнить логин и пароль и выбрать роль. Данные отправляются на сервер. Там они проверяются на корректность и на наличие такого пользователя. Если такой пользователь существует, то приходит положительный ответ от сервера и происходит переход в главное меню.

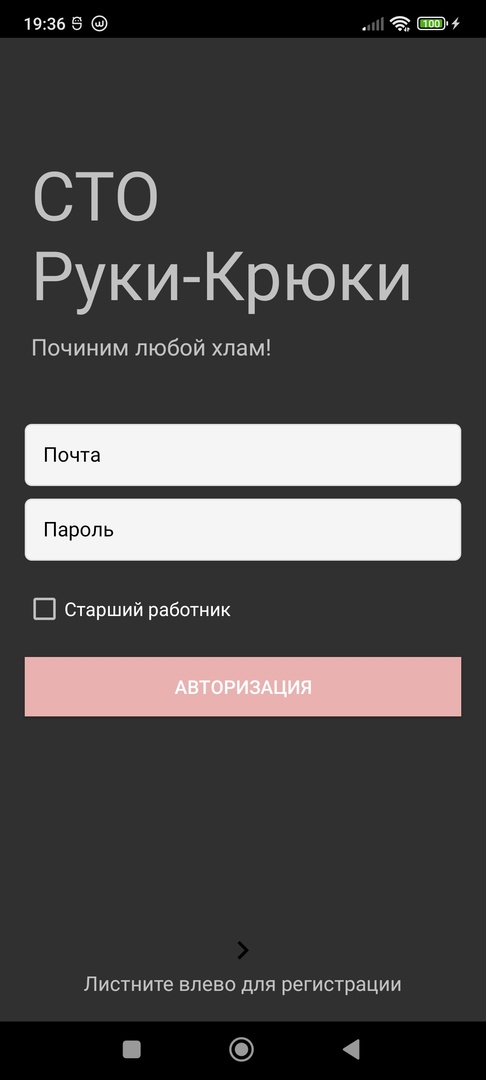


Рисунок 18. Авторизация

Главное меню

Показывает список ТО, созданных работником. Они подгружаются с сервера. Можно перемещаться между фрагментами с помощью нижнего меню. Содержит кнопку для создания нового ТО и перехода на такой фрагмент. Если долго нажимать на элемент, то происходит переход к фрагменту просмотра ТО.

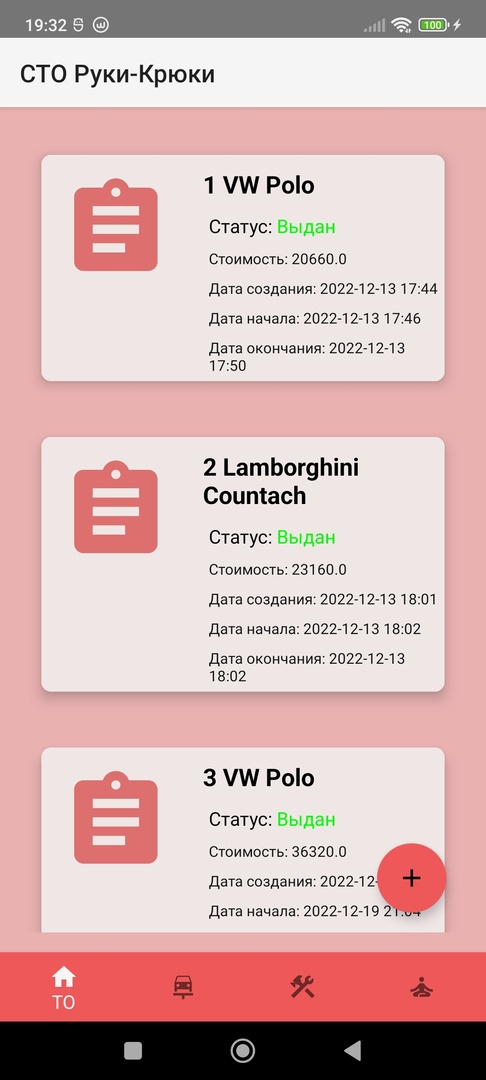


Рисунок 19. Главное меню

Создание ТО.

Подгружаются данные о существующих автомобилях. Необходимо выбрать автомобиль. При нажатии на кнопку “Создать” происходит сохранение ТО. Посылается запрос на создание. Если данные корректны, то приходит положительный ответ и данные записываются в на сервер в хранилище.

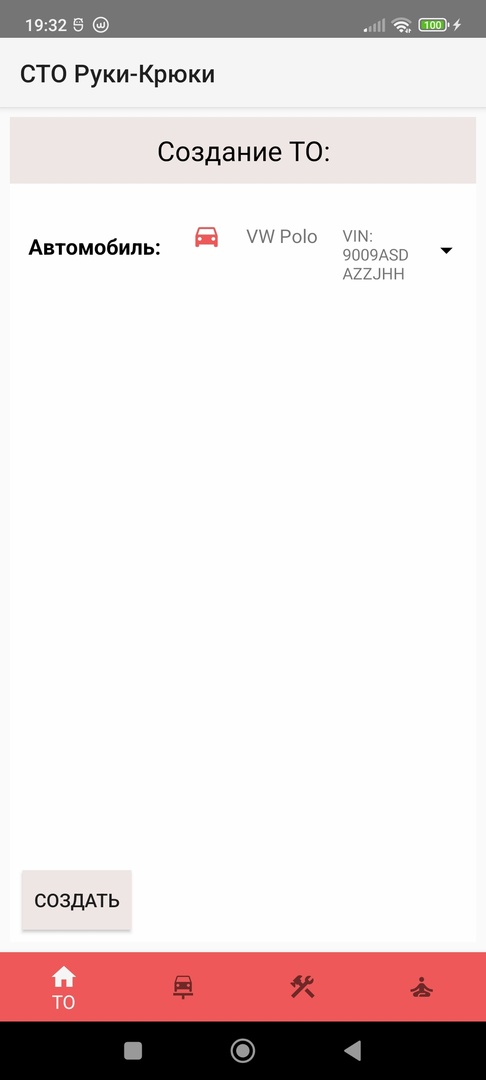


Рисунок 20. Создание ТО.

Просмотр ТО.

При длительном нажатии на ТО, происходит переход в этот фрагмент. Подгружаются данные с сервера об этом ТО. Можно изменить статус ТО в соответствии с его текущим статусом. Если такое сделать, то произойдет запрос на сервер, который, в случае корректных данных изменит статус ТО, иначе придет отрицательный ответ и ничего с данными не произойдёт. Видно услуги, которые включены в ТО.

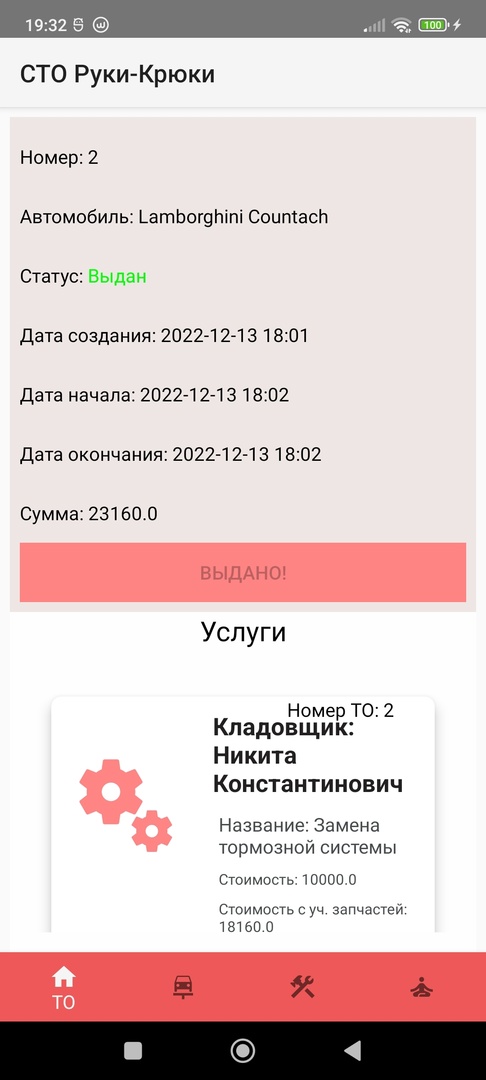


Рисунок 21. Просмотр ТО.

Фрагмент просмотра автомобилей.

При нажатии на вторую иконку нижнего меню происходит переход в этот фрагмент. Подгружаются данные с сервера о существующих в системе авто. Здесь можно увидеть текущих клиентов СТО. При нажатии на кнопку добавления, происходит переход в фрагмент добавления нового клиента. При длительном нажатии на пункт авто происходит переход в фрагмент просмотра авто и его записей.

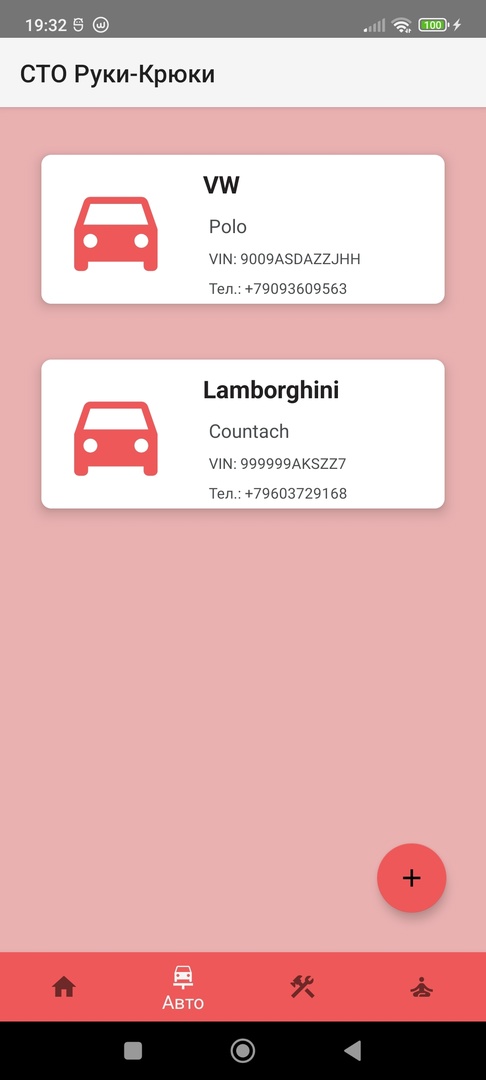


Рисунок 22. Просмотр авто

Фрагмент добавления авто.

При нажатии на кнопку происходит переход в этом фрагмент. При нажатии на кнопку “Добавить” происходит отправка запроса о создании новой записи на сервер, если данные корректны, то придёт положительный ответ и запись создастся, иначе ничего не произойдет. Произойдет переход на предыдущий фрагмент. Необходимо заполнить:

1. Марку
2. Модель
3. VIN
4. Номер телефона владельца

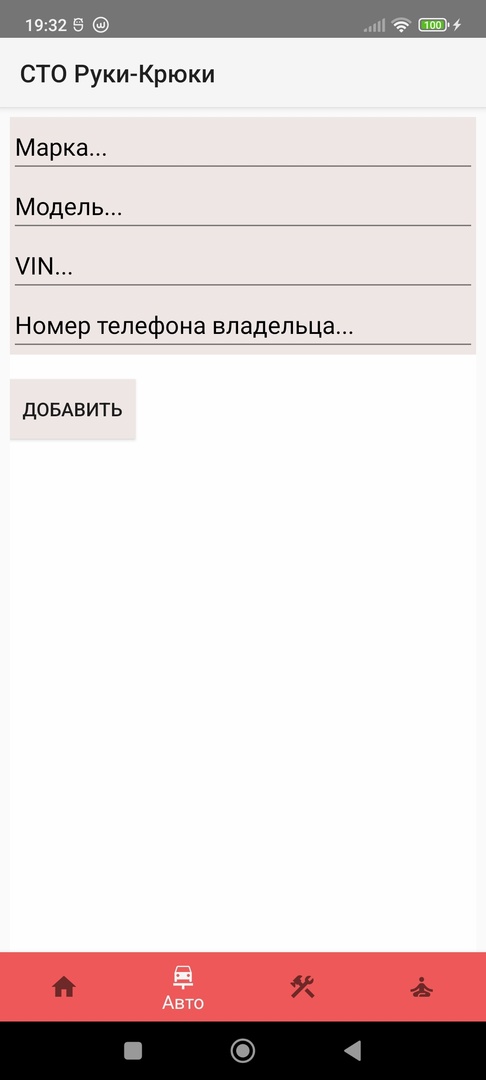


Рисунок 23. Добавление авто

Фрагмент просмотра информации об авто.

При длительном нажатии на пункт авто происходит переход в этот фрагмент. Подгружаются данные об этом автомобиле. Поля заполняются данными. Показываются записи сервисов данного авто. При длительном нажатии на пункт записи сервисов происходит переход на фрагмент редактирования записи сервисов. Либо при нажатии на кнопку “Изменить” произойдет запрос на сервер об изменении существующей сущности, если данные корректны, то положительный ответ и сущности обновиться, иначе ничего не произойдёт.

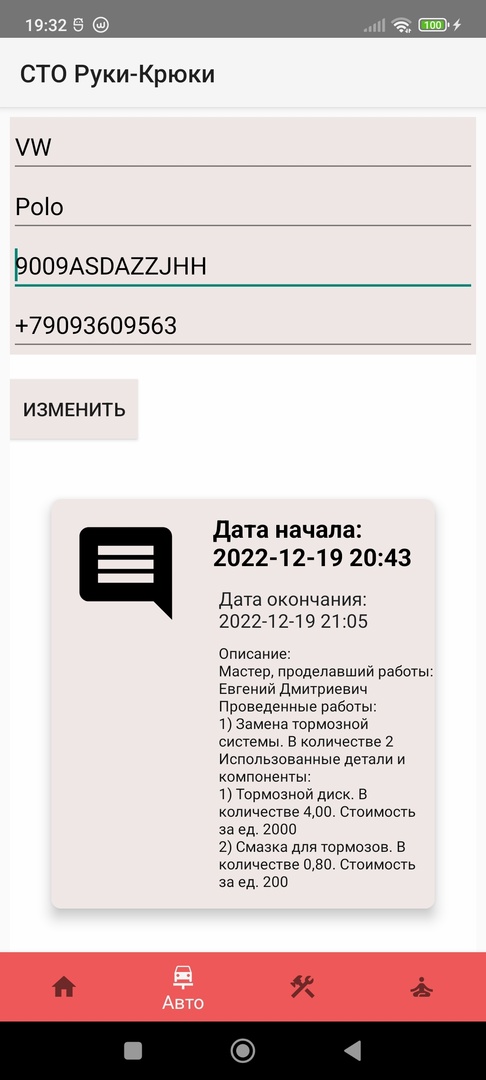


Рисунок 24. Редактирование авто

Фрагмент редактирования записи сервисов.

При длительном нажатии на пункт записи сервисов происходит переход на этот фрагмент. Данные о записях подгружаются с сервера. Поля заполняются данными. Можно изменять описание. При нажатии на кнопку “Изменить” происходит запрос на сохранения внесенных изменений, если данные корректны, то придёт положительный ответ и запись обновится, иначе ничего не произойдет. Произойдет переход на предыдущий фрагмент.

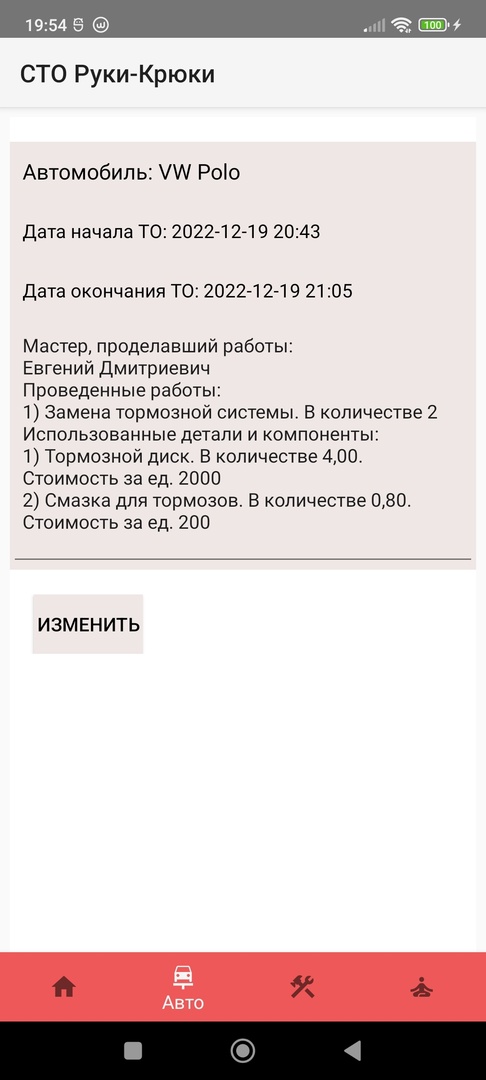


Рисунок 25. Изменение записи сервиса

Фрагмент просмотра услуг.

При нажатии на третий пункт меню происходит переход на этот фрагмент. Подгружаются данные с сервера об услугах. При нажатии на кнопку происходит переход на фрагмент создания услуги. При длительном нажатии на услугу происходит переход на фрагмент просмотра информации о ней.

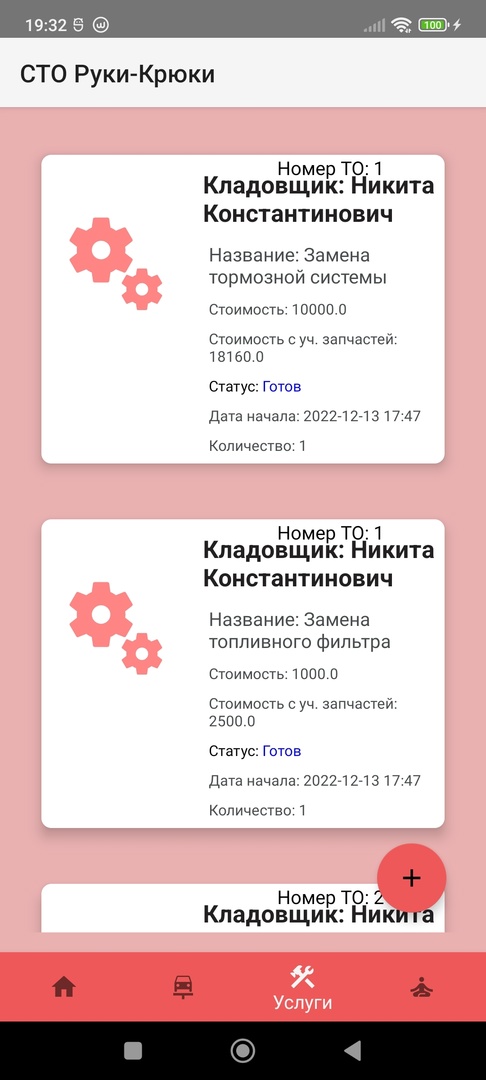


Рисунок 26. Услуги

Фрагмент создания услуги. Необходимо заполнить требуемые поля и выбрать ТО, тип услуги и кладовщика. При вводе количество будет подсчитана стоимость работы. При нажатии на кнопку добавить произойдет запрос на сервер на создание услуги, если данные корректны, то придёт положительный ответ и данные сохранятся, иначе ничего не произойдет. Произойдет переход на предыдущий фрагмент.

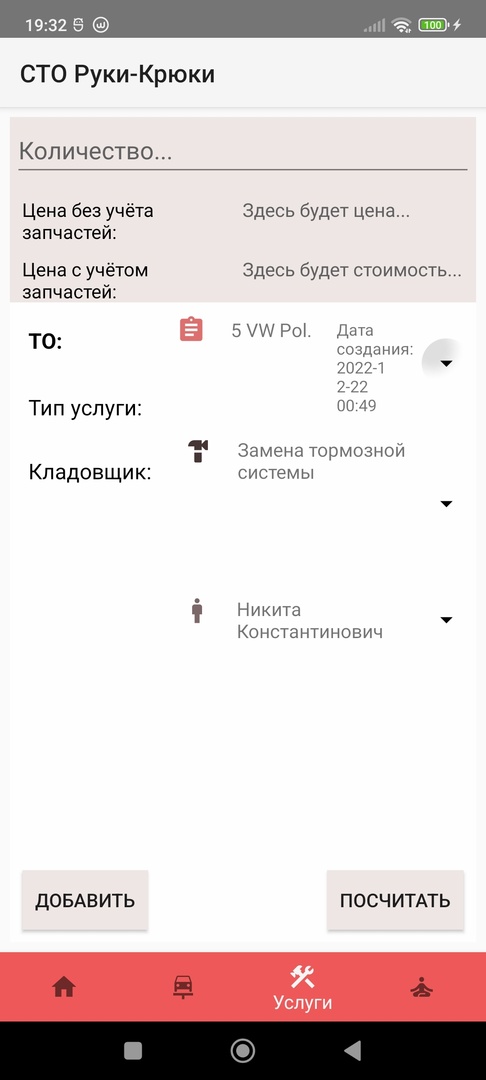


Рисунок 27. Создание услуги

Фрагмент просмотра информации об услуге.

При длительном нажатии на пункт услуги происходит переход на данный фрагмент. Подгружаются данные об услуге с сервера. Показывается информация об услуге, требуемые для нее запчасти и её статус.

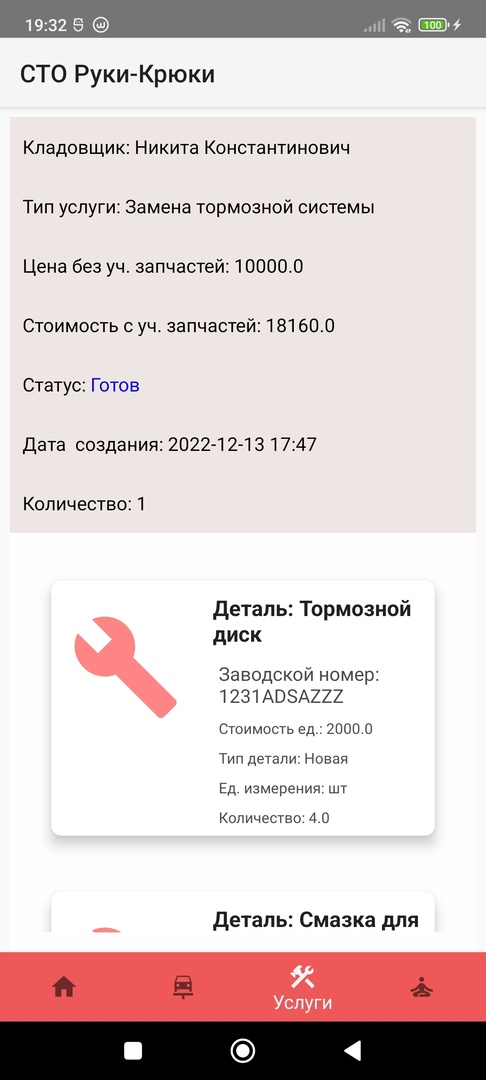


Рисунок 28. Просмотр услуги

При нажатии на четвертый пункт меню происходит переход во фрагмент данных о профиле. Подгружаются данные об аккаунте с сервера. Здесь показаны данные вашего аккаунта. При нажатии на кнопку “Изменить данные” произойдет запрос на изменение данных об аккаунте, если данные корректны, то придёт положительный ответ и данные изменятся, иначе ничего не произойдет. При нажатии на кнопку “Отчёт по услугам”, создастся соответствующий отчёт. При нажатии на кнопку “Отчёт по автомобилю” и выборе соответствующего авто создастся соответствующий отчёт об автомобиле. При нажатии на кнопку “Отчёт по проделанным ТО” создастся соответствующий отчёт о проведенных ТО. При нажатии на кнопку “Выйти из аккаунта” произойдет переход в активити авторизации.

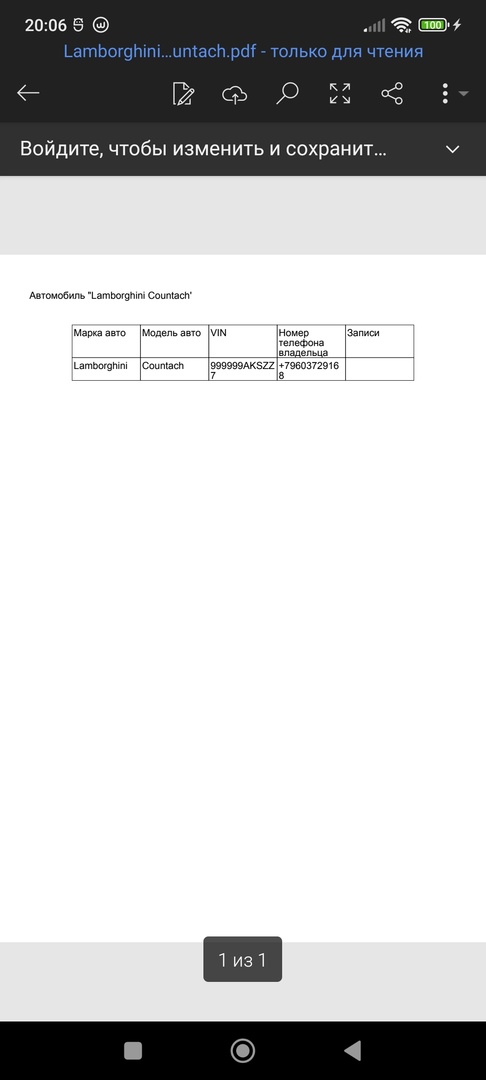


Рисунок 29. Отчёт об авто

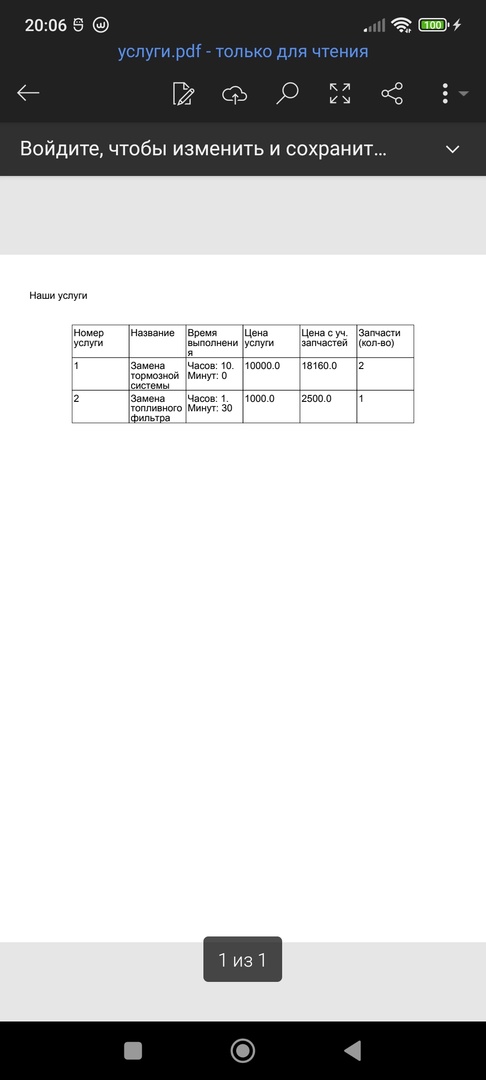


Рисунок 30. Отчёт об услугах

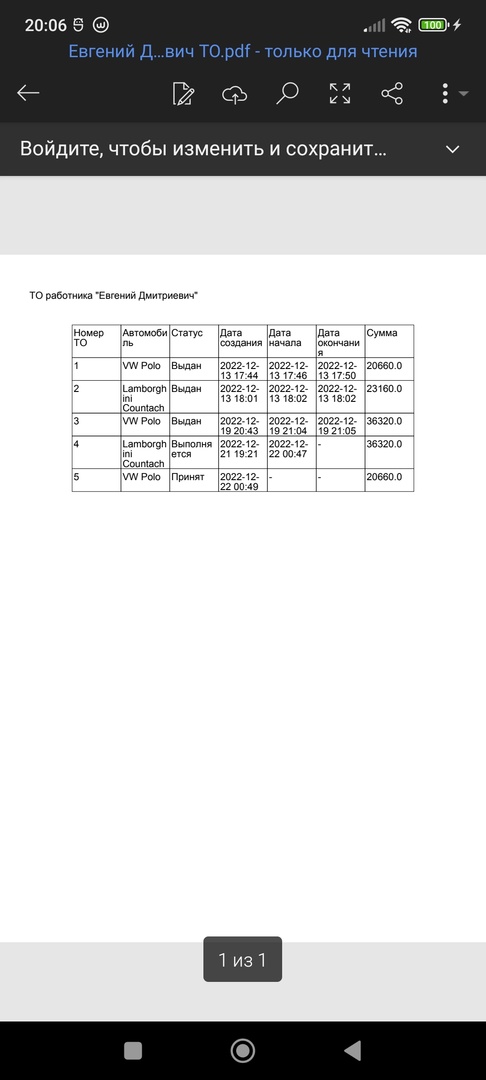


Рисунок 31. Отчёт о проведенных ТО работником

# **Заключение**

В процессе создания курсовой работы была проанализирована предметная область, были использованы возможности среды разработки Android Studio, возможности системы Android, создан дизайн интерфейса, продуманы возможные действия пользователя, созданы диаграммы, созданы шаблоны возможных отчетов, разработан программный продукт.

Также были улучшено знание языка Java под Android, улучшено понимание Asp .Net Core и языка C#. Была изучена возможность работы с сервером через библиотеку Retrofit2, было улучшено знание о сериализации и десериализации с помощью библиотеки GSON. Получено знание о взаимодействии с сервером через клиент на другом устройстве, произведена работа по создания файлов формата PDF.

# **Список литературы**

1. Общие сведения об ASP.NET Core [Электронный ресурс] / Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0
2. Адаптеры и списки [Электронный ресурс] / Режим доступа:<https://metanit.com/java/android/5.1.php>
3. StartAndroid: Активити [Электронный ресурс] / Режим доступа:

https://startandroid.ru/ru/uroki/vse-uroki-spiskom/58-urok-21-sozdanie-i-vyzov-activity.html

1. Spinner – выпадающий список [Электронный ресурс] / Режим доступа:<https://startandroid.ru/ru/uroki/vse-uroki-spiskom/115-urok-56-spinner-vypadajuschij-spisok.html>
2. Работа с файловой системой Чтение и сохранение файлов [Электронный ресурс] / Режим доступа:

https://metanit.com/java/android/13.1.php

1. Java – Read and Write PDF with iText] [Электронный ресурс] / Режим доступа: <https://howtodoinjava.com/java/library/read-generate-pdf-java-itext/>
2. Retrofit – [Электронный ресурс] / Режим доступа:

https://square.github.io/retrofit/

# **Приложение**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <uses-permission android:name="android.permission.INTERNET" />  
 <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
 <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
  
 <application  
 android:name=".App"  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MyApplication"  
 android:usesCleartextTraffic="true"  
 tools:targetApi="31">  
 <activity  
 android:name=".LogInRegisterActivity"  
 android:exported="true"  
 android:theme="@style/Theme.MyApplication.loging">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 <meta-data  
 android:name="android.app.lib\_name"  
 android:value="" />  
 </activity>  
 <activity  
 android:name=".MainActivity"  
 android:exported="false">  
 <meta-data  
 android:name="android.app.lib\_name"  
 android:value="" />  
 </activity>  
 </application>  
  
</manifest>

package com.example.myapplication.adapters;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.CarViewModel;  
  
import java.util.ArrayList;  
  
public class CarListAdapter extends RecyclerView.Adapter<CarListAdapter.CarViewHolder> {  
 private Context context;  
 private ArrayList<CarViewModel> cars;  
 //private ArrayList<CarViewModel> cars\_selected;  
 private OnItemLongClickListener listener;  
 private final String VIN = "VIN: ";  
 private final String Phone = "Тел.: ";  
 public CarListAdapter(Context c) {  
 this.context = c;  
 cars = new ArrayList<>();  
 }  
  
 public void filterList(ArrayList<CarViewModel> filterList) {  
 cars = filterList;  
 notifyDataSetChanged();  
 }  
  
 public void setData(ArrayList<CarViewModel> cars) {  
 this.cars = cars;  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull CarViewHolder holder, int position) {  
 if (cars != null && cars.size() > 0) {  
 CarViewModel car = cars.get(position);  
 holder.brandTV.setText(car.getBrand());  
 holder.modelTV.setText(car.getModel());  
 holder.vinTV.setText(VIN + car.getVIN());  
 holder.phoneTV.setText(Phone + car.getOwnerPhoneNumber());  
 }  
 }  
  
 @Override  
 public int getItemCount() {  
 if (cars != null)  
 return cars.size();  
 return 0;  
 }  
  
 public CarViewModel getCarAt(int position) {  
 return cars.get(position);  
 }  
  
 @NonNull  
 @Override  
 public CarViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.car\_item\_recyclerview, parent, false);  
 return new CarViewHolder(view);  
 }  
  
 class CarViewHolder extends RecyclerView.ViewHolder{  
 private final TextView brandTV;  
 private final TextView modelTV;  
 private final TextView vinTV;  
 private final TextView phoneTV;  
  
 private CarViewHolder(View itemView) {  
 super(itemView);  
 brandTV = itemView.findViewById(R.id.work\_item\_storekeeperTV);  
 modelTV = itemView.findViewById(R.id.work\_item\_nameTV);  
 vinTV = itemView.findViewById(R.id.work\_item\_priceTV);  
 phoneTV = itemView.findViewById(R.id.work\_item\_netpriceTV);  
 itemView.setOnLongClickListener(new View.OnLongClickListener() {  
 @Override  
 public boolean onLongClick(View v) {  
 int position = getAdapterPosition();  
 if (listener != null && position != RecyclerView.NO\_POSITION)  
 listener.onItemLongClicked(cars.get(position), position);  
 return true;  
 }  
 });  
 }  
 }  
  
 public interface OnItemLongClickListener {  
 void onItemLongClicked(CarViewModel car, int position);  
 }  
  
 public void setOnItemLongClickListener(OnItemLongClickListener listener) {  
 this.listener = listener;  
 }  
}

package com.example.myapplication.adapters;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.ServiceRecordViewModel;  
  
import java.util.ArrayList;  
  
public class RecordListAdapter extends RecyclerView.Adapter<RecordListAdapter.RecordViewHolder> {  
 private Context context;  
 private ArrayList<ServiceRecordViewModel> records;  
 private OnItemLongClickListener listener;  
 private final String DateB = "Дата начала: ";  
 private final String DateE = "Дата окончания: ";  
 private final String Description = "Описание:\n";  
  
 public RecordListAdapter(Context c) {  
 this.context = c;  
 records = new ArrayList<>();  
 }  
  
 public void filterList(ArrayList<ServiceRecordViewModel> filterList) {  
 records = filterList;  
 notifyDataSetChanged();  
 }  
  
 public void setData(ArrayList<ServiceRecordViewModel> records) {  
 this.records = records;  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull RecordViewHolder holder, int position) {  
 if (records != null && records.size() > 0) {  
 ServiceRecordViewModel record = records.get(position);  
 String date = record.getDateBegin().split("T")[0] + " " + record.getDateBegin().split("T")[1].substring(0, 5);  
 holder.dateBeginTV.setText(DateB + date);  
 date = record.getDateEnd().split("T")[0] + " " + record.getDateEnd().split("T")[1].substring(0, 5);  
 holder.dateEndTV.setText(DateE + date);  
 holder.descriptionTV.setText(Description + record.getDescription());  
 }  
 }  
  
 @Override  
 public int getItemCount() {  
 if (records != null)  
 return records.size();  
 return 0;  
 }  
  
 @NonNull  
 @Override  
 public RecordViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.record\_item\_recyclerview, parent, false);  
 return new RecordViewHolder(view);  
 }  
  
 class RecordViewHolder extends RecyclerView.ViewHolder {  
 private final TextView dateBeginTV;  
 private final TextView dateEndTV;  
 private final TextView descriptionTV;  
  
 private RecordViewHolder(View itemView) {  
 super(itemView);  
 dateBeginTV = itemView.findViewById(R.id.to\_to\_and\_carNameTV);  
 dateEndTV = itemView.findViewById(R.id.to\_statusTV);  
 descriptionTV = itemView.findViewById(R.id.to\_sumTV);  
  
 itemView.setOnLongClickListener(new View.OnLongClickListener() {  
 @Override  
 public boolean onLongClick(View v) {  
 int position = getAdapterPosition();  
 if (listener != null && position != RecyclerView.NO\_POSITION)  
 listener.onItemLongClicked(records.get(position), position);  
 return true;  
 }  
 });  
 }  
 }  
  
 public interface OnItemLongClickListener {  
 void onItemLongClicked(ServiceRecordViewModel record, int position);  
 }  
  
 public void setOnItemLongClickListener(OnItemLongClickListener listener) {  
 this.listener = listener;  
 }  
}

package com.example.myapplication.adapters;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.CarViewModel;  
import com.example.myapplication.models.viewModels.SparePartViewModel;  
import com.example.myapplication.models.webViewModels.SparePartWebViewModel;  
  
import java.util.ArrayList;  
  
public class SparePartListAdapter extends RecyclerView.Adapter<SparePartListAdapter.SparePartViewHolder> {  
 private Context context;  
 private ArrayList<SparePartWebViewModel> parts;  
  
 public SparePartListAdapter(Context c) {  
 this.context = c;  
 parts = new ArrayList<>();  
 }  
  
 public void filterList(ArrayList<SparePartWebViewModel> filterList) {  
 parts = filterList;  
 notifyDataSetChanged();  
 }  
  
 public void setData(ArrayList<SparePartWebViewModel> cars) {  
 this.parts = cars;  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull SparePartViewHolder holder, int position) {  
 if (parts != null && parts.size() > 0) {  
 SparePartWebViewModel part = parts.get(position);  
 holder.sparepart\_item\_nameTV.setText("Деталь: " + part.getName());  
 holder.sparepart\_item\_factorynumberTV.setText("Заводской номер: " + part.getFactoryNumber());  
 holder.sparepart\_item\_priceTV.setText("Стоимость ед.: " + part.getPrice());  
 holder.sparepart\_item\_typeTV.setText("Тип детали: " + part.getType());  
 holder.sparepart\_item\_measurementTV.setText("Ед. измерения: " + part.getUMeasurement());  
 holder.sparepart\_item\_countTV.setText("Количество: " + part.getCount());  
 }  
 }  
  
 @Override  
 public int getItemCount() {  
 if (parts != null)  
 return parts.size();  
 return 0;  
 }  
  
 @NonNull  
 @Override  
 public SparePartViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.sparepart\_item\_recyclerview, parent, false);  
 return new SparePartViewHolder(view);  
 }  
  
 class SparePartViewHolder extends RecyclerView.ViewHolder{  
 private final TextView sparepart\_item\_nameTV;  
 private final TextView sparepart\_item\_factorynumberTV;  
 private final TextView sparepart\_item\_priceTV;  
 private final TextView sparepart\_item\_typeTV;  
 private final TextView sparepart\_item\_measurementTV;  
 private final TextView sparepart\_item\_countTV;  
  
 private SparePartViewHolder(View itemView) {  
 super(itemView);  
 sparepart\_item\_nameTV = itemView.findViewById(R.id.sparepart\_item\_nameTV);  
 sparepart\_item\_factorynumberTV = itemView.findViewById(R.id.sparepart\_item\_factorynumberTV);  
 sparepart\_item\_priceTV = itemView.findViewById(R.id.sparepart\_item\_priceTV);  
 sparepart\_item\_typeTV = itemView.findViewById(R.id.sparepart\_item\_typeTV);  
 sparepart\_item\_measurementTV = itemView.findViewById(R.id.sparepart\_item\_measurementTV);  
 sparepart\_item\_countTV = itemView.findViewById(R.id.sparepart\_item\_countTV);  
 }  
 }  
}

package com.example.myapplication.adapters;  
  
import android.content.Context;  
import android.graphics.Color;  
import android.text.Html;  
import android.text.SpannableString;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.CarViewModel;  
import com.example.myapplication.models.viewModels.TOViewModel;  
import com.example.myapplication.enums.TOStatus;  
  
import java.util.ArrayList;  
  
public class TOListAdapter extends RecyclerView.Adapter<TOListAdapter.TOViewHolder> {  
 private Context context;  
 private ArrayList<TOViewModel> tos;  
 private OnItemLongClickListener listener;  
  
 public TOListAdapter(Context c) {  
 this.context = c;  
 tos = new ArrayList<>();  
 //cars\_selected = new ArrayList<>();  
 }  
  
 public void filterList(ArrayList<TOViewModel> filterList) {  
 tos = filterList;  
 notifyDataSetChanged();  
 }  
  
 public void setData(ArrayList<TOViewModel> tos) {  
 this.tos = tos;  
 }  
  
// public ArrayList<CarViewModel> getSelected\_cars() {  
// return cars\_selected;  
// }  
  
 @Override  
 public void onBindViewHolder(@NonNull TOViewHolder holder, int position) {  
 if (tos != null && tos.size() > 0) {  
 TOViewModel to = tos.get(position);  
 holder.to\_to\_and\_carNameTV.setText(to.getTOAndCarName());  
 String status = to.getStatus();  
 holder.to\_statusTV.setText(Html.fromHtml(getStatus(status)), TextView.BufferType.SPANNABLE);  
 holder.to\_sumTV.setText("Стоимость: " + to.getSum());  
 holder.to\_dateCreateTV.setText("Дата создания: " + getDateNormalFormat(to.getDateCreate()));  
 holder.to\_dateImplementTV.setText("Дата начала: " + getDateNormalFormat(to.getDateImplement()));  
 holder.to\_dateOverTV.setText("Дата окончания: " + getDateNormalFormat(to.getDateOver()));  
 }  
 }  
  
 private String getDateNormalFormat(String s) {  
 if (!s.equals("-"))  
 return s.split("T")[0] + " " + s.split("T")[1].substring(0, 5);  
 return "-";  
 }  
  
 private String getStatus(String s) {  
 switch (s) {  
 case "Принят":  
 return "<font color='black'>Статус: </font><font color='red'>" + s + "</font>";  
 case "Выполняется":  
 return "<font color='black'>Статус: </font><font color='yellow'>" + s + "</font>";  
 case "Готов":  
 return "<font color='black'>Статус: </font><font color='blue'>" + s + "</font>";  
 default:  
 return "<font color='black'>Статус: </font><font color='green'>" + s + "</font>";  
 }  
 }  
  
 @Override  
 public int getItemCount() {  
 if (tos != null)  
 return tos.size();  
 return 0;  
 }  
  
 @NonNull  
 @Override  
 public TOViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.to\_item\_recyclerview, parent, false);  
 return new TOViewHolder(view);  
 }  
  
 class TOViewHolder extends RecyclerView.ViewHolder {  
 private final TextView to\_to\_and\_carNameTV;  
 private final TextView to\_statusTV;  
 private final TextView to\_sumTV;  
 private final TextView to\_dateCreateTV;  
 private final TextView to\_dateImplementTV;  
 private final TextView to\_dateOverTV;  
  
 private TOViewHolder(View itemView) {  
 super(itemView);  
 to\_to\_and\_carNameTV = itemView.findViewById(R.id.to\_to\_and\_carNameTV);  
 to\_statusTV = itemView.findViewById(R.id.to\_statusTV);  
 to\_sumTV = itemView.findViewById(R.id.to\_sumTV);  
 to\_dateCreateTV = itemView.findViewById(R.id.to\_dateCreateTV);  
 to\_dateImplementTV = itemView.findViewById(R.id.to\_dateImplementTV);  
 to\_dateOverTV = itemView.findViewById(R.id.to\_dateOverTV);  
  
 itemView.setOnLongClickListener(new View.OnLongClickListener() {  
 @Override  
 public boolean onLongClick(View v) {  
 int position = getAdapterPosition();  
 if (listener != null && position != RecyclerView.NO\_POSITION)  
 listener.onItemLongClicked(tos.get(position), position);  
 return true;  
 }  
 });  
 }  
 }  
  
 public interface OnItemLongClickListener {  
 void onItemLongClicked(TOViewModel to, int position);  
 }  
  
 public void setOnItemLongClickListener(OnItemLongClickListener listener) {  
 this.listener = listener;  
 }  
}

package com.example.myapplication.adapters;  
  
import android.content.Context;  
import android.text.Html;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.TOViewModel;  
import com.example.myapplication.models.viewModels.WorkViewModel;  
  
import java.util.ArrayList;  
  
public class WorkListAdapter extends RecyclerView.Adapter<WorkListAdapter.WorkViewHolder> {  
 private Context context;  
 private ArrayList<WorkViewModel> works;  
 private OnItemLongClickListener listener;  
  
 public WorkListAdapter(Context c) {  
 this.context = c;  
 works = new ArrayList<>();  
 }  
  
 public void filterList(ArrayList<WorkViewModel> filterList) {  
 works = filterList;  
 notifyDataSetChanged();  
 }  
  
 public void setData(ArrayList<WorkViewModel> works) {  
 this.works = works;  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull WorkViewHolder holder, int position) {  
 if (works != null && works.size() > 0) {  
 WorkViewModel work = works.get(position);  
 holder.work\_item\_toIdTV.setText("Номер ТО: " + work.getTOId());  
 holder.work\_item\_storekeeperTV.setText("Кладовщик: " + work.getStoreKeeperFIO());  
 holder.work\_item\_nameTV.setText("Название: " + work.getWorkName());  
 holder.work\_item\_priceTV.setText("Стоимость: " + work.getPrice());  
 holder.work\_item\_netpriceTV.setText("Стоимость с уч. запчастей: " + work.getNetPrice());  
 String status = work.getWorkStatus();  
 holder.work\_item\_statusTV.setText(Html.fromHtml(getStatus(status)), TextView.BufferType.SPANNABLE);  
 holder.work\_item\_datebeginTV.setText("Дата начала: " + getDateNormalFormat(work.getWorkBegin()));  
 holder.work\_item\_countTV.setText("Количество: " + work.getCount());  
 }  
 }  
  
 private String getDateNormalFormat(String s) {  
 if (!s.equals("-"))  
 return s.split("T")[0] + " " + s.split("T")[1].substring(0, 5);  
 return "-";  
 }  
  
 private String getStatus(String s) {  
 switch (s) {  
 case "Принят":  
 return "<font color='black'>Статус: </font><font color='red'>" + s + "</font>";  
 case "Выполняется":  
 return "<font color='black'>Статус: </font><font color='yellow'>" + s + "</font>";  
 default:  
 return "<font color='black'>Статус: </font><font color='blue'>" + s + "</font>";  
 }  
 }  
  
 @Override  
 public int getItemCount() {  
 if (works != null)  
 return works.size();  
 return 0;  
 }  
  
 @NonNull  
 @Override  
 public WorkViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.work\_item\_recyclerview, parent, false);  
 return new WorkViewHolder(view);  
 }  
  
 class WorkViewHolder extends RecyclerView.ViewHolder {  
 private final TextView work\_item\_storekeeperTV;  
 private final TextView work\_item\_nameTV;  
 private final TextView work\_item\_priceTV;  
 private final TextView work\_item\_netpriceTV;  
 private final TextView work\_item\_statusTV;  
 private final TextView work\_item\_datebeginTV;  
 private final TextView work\_item\_countTV;  
 private final TextView work\_item\_toIdTV;  
  
 private WorkViewHolder(View itemView) {  
 super(itemView);  
 work\_item\_storekeeperTV = itemView.findViewById(R.id.work\_item\_storekeeperTV);  
 work\_item\_nameTV = itemView.findViewById(R.id.work\_item\_nameTV);  
 work\_item\_priceTV = itemView.findViewById(R.id.work\_item\_priceTV);  
 work\_item\_netpriceTV = itemView.findViewById(R.id.work\_item\_netpriceTV);  
 work\_item\_statusTV = itemView.findViewById(R.id.work\_item\_statusTV);  
 work\_item\_datebeginTV = itemView.findViewById(R.id.work\_item\_datebeginTV);  
 work\_item\_countTV = itemView.findViewById(R.id.work\_item\_countTV);  
 work\_item\_toIdTV = itemView.findViewById(R.id.work\_item\_toIdTV);  
  
 if (listener != null) {  
 itemView.setOnLongClickListener(new View.OnLongClickListener() {  
 @Override  
 public boolean onLongClick(View v) {  
 int position = getAdapterPosition();  
 if (listener != null && position != RecyclerView.NO\_POSITION)  
 listener.onItemLongClicked(works.get(position), position);  
 return true;  
 }  
 });  
 }  
 }  
 }  
  
 public interface OnItemLongClickListener {  
 void onItemLongClicked(WorkViewModel work, int position);  
 }  
  
 public void setOnItemLongClickListener(OnItemLongClickListener listener) {  
 this.listener = listener;  
 }  
}

package com.example.myapplication.enums;  
  
public enum TOStatus {  
 Принят,  
  
 Выполняется,  
  
 Готов,  
  
 Выдан  
}

package com.example.myapplication.enums;  
  
public enum WorkStatus {  
 Принят,  
  
 Выполняется,  
  
 Готов  
}

package com.example.myapplication.models.bindingModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
import java.util.Map;  
  
public class CarBindingModel {  
 @SerializedName("id")  
 @Expose  
 public Integer Id;  
  
 @SerializedName("brand")  
 @Expose  
 public String Brand;  
  
  
 @SerializedName("model")  
 @Expose  
 public String Model;  
  
 @SerializedName("vin")  
 @Expose  
 public String VIN;  
  
  
 @SerializedName("ownerPhoneNumber")  
 @Expose  
 public String OwnerPhoneNumber;  
  
 @SerializedName("records")  
 @Expose  
 public Map<Integer, String> Records;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public String getBrand() {  
 return Brand;  
 }  
  
 public void setBrand(String brand) {  
 Brand = brand;  
 }  
  
 public String getModel() {  
 return Model;  
 }  
  
 public void setModel(String model) {  
 Model = model;  
 }  
  
 public String getVIN() {  
 return VIN;  
 }  
  
 public void setVIN(String VIN) {  
 this.VIN = VIN;  
 }  
  
 public String getOwnerPhoneNumber() {  
 return OwnerPhoneNumber;  
 }  
  
 public void setOwnerPhoneNumber(String ownerPhoneNumber) {  
 OwnerPhoneNumber = ownerPhoneNumber;  
 }  
  
 public Map<Integer, String> getRecords() {  
 return Records;  
 }  
  
 public void setRecords(Map<Integer, String> records) {  
 Records = records;  
 }  
}

package com.example.myapplication.models.bindingModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class ChangeTOStatusBindingModel {  
 @SerializedName("toId")  
 @Expose  
 public Integer TOId;  
  
 public Integer getTOid() {  
 return TOId;  
 }  
  
 public void setTOid(Integer TOId) {  
 this.TOId = TOId;  
 }  
}

package com.example.myapplication.models.bindingModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class CreateTOBindingModel {  
 @SerializedName("carId")  
 @Expose  
 public Integer CarId;  
  
 @SerializedName("employeeId")  
 @Expose  
 public Integer EmployeeId;  
  
 public Integer getCarId() {  
 return CarId;  
 }  
  
 public void setCarId(Integer carId) {  
 CarId = carId;  
 }  
  
 public Integer getEmployeeId() {  
 return EmployeeId;  
 }  
  
 public void setEmployeeId(Integer employeeId) {  
 EmployeeId = employeeId;  
 }  
  
 public Float getSum() {  
 return Sum;  
 }  
  
 public void setSum(Float sum) {  
 Sum = sum;  
 }  
  
 @SerializedName("sum")  
 @Expose  
 public Float Sum;  
}

package com.example.myapplication.models.bindingModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class CreateWorkBindingModel {  
 @SerializedName("storeKeeperId")  
 @Expose  
 public Integer StoreKeeperId;  
  
 @SerializedName("workTypeId")  
 @Expose  
 public Integer WorkTypeId;  
  
 @SerializedName("toId")  
 @Expose  
 public Integer TOId;  
  
 @SerializedName("count")  
 @Expose  
 public Integer Count;  
  
 public Integer getStoreKeeperId() {  
 return StoreKeeperId;  
 }  
  
 public void setStoreKeeperId(Integer storeKeeperId) {  
 StoreKeeperId = storeKeeperId;  
 }  
  
 public Integer getWorkTypeId() {  
 return WorkTypeId;  
 }  
  
 public void setWorkTypeId(Integer workTypeId) {  
 WorkTypeId = workTypeId;  
 }  
  
 public Integer getTOId() {  
 return TOId;  
 }  
  
 public void setTOId(Integer TOId) {  
 this.TOId = TOId;  
 }  
  
 public Integer getCount() {  
 return Count;  
 }  
  
 public void setCount(Integer count) {  
 Count = count;  
 }  
  
 public Float getPrice() {  
 return Price;  
 }  
  
 public void setPrice(Float price) {  
 Price = price;  
 }  
  
 public Float getNetPrice() {  
 return NetPrice;  
 }  
  
 public void setNetPrice(Float netPrice) {  
 NetPrice = netPrice;  
 }  
  
 @SerializedName("price")  
 @Expose  
 public Float Price;  
  
 @SerializedName("netPrice")  
 @Expose  
 public Float NetPrice;  
}

package com.example.myapplication.models.bindingModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class EmployeeBindingModel {  
 @SerializedName("id")  
 @Expose  
 public Integer Id;  
  
 @SerializedName("fio")  
 @Expose  
 public String FIO;  
  
 @SerializedName("login")  
 @Expose  
 public String Login;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public String getFIO() {  
 return FIO;  
 }  
  
 public void setFIO(String FIO) {  
 this.FIO = FIO;  
 }  
  
 public String getLogin() {  
 return Login;  
 }  
  
 public void setLogin(String login) {  
 Login = login;  
 }  
  
 public String getPassword() {  
 return Password;  
 }  
  
 public void setPassword(String password) {  
 Password = password;  
 }  
  
 @SerializedName("password")  
 @Expose  
 public String Password;  
}

package com.example.myapplication.models.bindingModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class ServiceRecordBindingModel {  
 @SerializedName("id")  
 @Expose  
 public Integer Id;  
  
 @SerializedName("carId")  
 @Expose  
 public Integer CarId;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public Integer getCarId() {  
 return CarId;  
 }  
  
 public void setCarId(Integer carId) {  
 CarId = carId;  
 }  
  
 public String getDateBegin() {  
 return DateBegin;  
 }  
  
 public void setDateBegin(String dateBegin) {  
 DateBegin = dateBegin;  
 }  
  
 public String getDateEnd() {  
 return DateEnd;  
 }  
  
 public void setDateEnd(String dateEnd) {  
 DateEnd = dateEnd;  
 }  
  
 public String getDescription() {  
 return Description;  
 }  
  
 public void setDescription(String description) {  
 Description = description;  
 }  
  
 @SerializedName("dateBegin")  
 @Expose  
 public String DateBegin;  
  
 @SerializedName("dateEnd")  
 @Expose  
 public String DateEnd;  
  
 @SerializedName("description")  
 @Expose  
 public String Description;  
}

package com.example.myapplication.models.bindingModels;  
  
import com.example.myapplication.enums.TOStatus;  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class TOBindingModel {  
 @SerializedName("id")  
 @Expose  
 public Integer Id;  
  
 @SerializedName("carId")  
 @Expose  
 public Integer CarId;  
  
 @SerializedName("employeeId")  
 @Expose  
 public Integer EmployeeId;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public Integer getCarId() {  
 return CarId;  
 }  
  
 public void setCarId(Integer carId) {  
 CarId = carId;  
 }  
  
 public Integer getEmployeeId() {  
 return EmployeeId;  
 }  
  
 public void setEmployeeId(Integer employeeId) {  
 EmployeeId = employeeId;  
 }  
  
 public Float getSum() {  
 return Sum;  
 }  
  
 public void setSum(Float sum) {  
 Sum = sum;  
 }  
  
 public TOStatus getStatus() {  
 return Status;  
 }  
  
 public void setStatus(TOStatus status) {  
 Status = status;  
 }  
  
 public String getDateCreate() {  
 return DateCreate;  
 }  
  
 public void setDateCreate(String dateCreate) {  
 DateCreate = dateCreate;  
 }  
  
 public String getDateImplement() {  
 return DateImplement;  
 }  
  
 public void setDateImplement(String dateImplement) {  
 DateImplement = dateImplement;  
 }  
  
 public String getDateOver() {  
 return DateOver;  
 }  
  
 public void setDateOver(String dateOver) {  
 DateOver = dateOver;  
 }  
  
 @SerializedName("sum")  
 @Expose  
 public Float Sum;  
  
 @SerializedName("status")  
 @Expose  
 public TOStatus Status;  
  
 @SerializedName("dateCreate")  
 @Expose  
 public String DateCreate;  
  
 @SerializedName("dateImplement")  
 @Expose  
 public String DateImplement;  
  
 @SerializedName("dateOver")  
 @Expose  
 public String DateOver;  
}

package com.example.myapplication.models.bindingModels;  
  
import com.example.myapplication.enums.WorkStatus;  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class WorkBindingModel {  
 @SerializedName("id")  
 @Expose  
 public Integer Id;  
  
 @SerializedName("storeKeeperId")  
 @Expose  
 public Integer StoreKeeperId;  
  
 @SerializedName("workTypeId")  
 @Expose  
 public Integer WorkTypeId;  
  
 @SerializedName("toId")  
 @Expose  
 public Integer TOId;  
  
 @SerializedName("workName")  
 @Expose  
 public String WorkName;  
  
 @SerializedName("price")  
 @Expose  
 public Float Price;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public Integer getStoreKeeperId() {  
 return StoreKeeperId;  
 }  
  
 public void setStoreKeeperId(Integer storeKeeperId) {  
 StoreKeeperId = storeKeeperId;  
 }  
  
 public Integer getWorkTypeId() {  
 return WorkTypeId;  
 }  
  
 public void setWorkTypeId(Integer workTypeId) {  
 WorkTypeId = workTypeId;  
 }  
  
 public Integer getTOId() {  
 return TOId;  
 }  
  
 public void setTOId(Integer TOId) {  
 this.TOId = TOId;  
 }  
  
 public String getWorkName() {  
 return WorkName;  
 }  
  
 public void setWorkName(String workName) {  
 WorkName = workName;  
 }  
  
 public Float getPrice() {  
 return Price;  
 }  
  
 public void setPrice(Float price) {  
 Price = price;  
 }  
  
 public Float getNetPrice() {  
 return NetPrice;  
 }  
  
 public void setNetPrice(Float netPrice) {  
 NetPrice = netPrice;  
 }  
  
 public WorkStatus getWorkStatus() {  
 return WorkStatus;  
 }  
  
 public void setWorkStatus(WorkStatus workStatus) {  
 WorkStatus = workStatus;  
 }  
  
 public String getWorkBegin() {  
 return WorkBegin;  
 }  
  
 public void setWorkBegin(String workBegin) {  
 WorkBegin = workBegin;  
 }  
  
 public Integer getCount() {  
 return Count;  
 }  
  
 public void setCount(Integer count) {  
 Count = count;  
 }  
  
 @SerializedName("netPrice")  
 @Expose  
 public Float NetPrice;  
  
 @SerializedName("workStatus")  
 @Expose  
 public WorkStatus WorkStatus;  
  
 @SerializedName("workBegin")  
 @Expose  
 public String WorkBegin;  
  
 @SerializedName("count")  
 @Expose  
 public Integer Count;  
}

package com.example.myapplication.models.viewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
import java.util.Map;  
  
public class CarViewModel {  
 @SerializedName("id")  
 @Expose  
 private Integer Id;  
  
 @SerializedName("brand")  
 @Expose  
 private String Brand;  
  
 public CarViewModel(String brand, String model, String VIN, String ownerPhoneNumber, Map<Integer, String> records) {  
 Brand = brand;  
 Model = model;  
 this.VIN = VIN;  
 OwnerPhoneNumber = ownerPhoneNumber;  
 Records = records;  
 }  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public String getBrand() {  
 return Brand;  
 }  
  
 public void setBrand(String brand) {  
 Brand = brand;  
 }  
  
 public String getModel() {  
 return Model;  
 }  
  
 public void setModel(String model) {  
 Model = model;  
 }  
  
 public String getVIN() {  
 return VIN;  
 }  
  
 public void setVIN(String VIN) {  
 this.VIN = VIN;  
 }  
  
 public String getOwnerPhoneNumber() {  
 return OwnerPhoneNumber;  
 }  
  
 public void setOwnerPhoneNumber(String ownerPhoneNumber) {  
 OwnerPhoneNumber = ownerPhoneNumber;  
 }  
  
 public Map<Integer, String> getRecords() {  
 return Records;  
 }  
  
 public void setRecords(Map<Integer, String> records) {  
 Records = records;  
 }  
  
 @SerializedName("model")  
 @Expose  
 private String Model;  
  
 @SerializedName("vin")  
 @Expose  
 private String VIN;  
  
 @SerializedName("ownerPhoneNumber")  
 @Expose  
 private String OwnerPhoneNumber;  
  
 /// <summary>  
 /// Записи сервисов (номер, ((Дата начала, Дата конца), описание)  
 /// </summary>  
 @SerializedName("records")  
 @Expose  
 private Map<Integer, String> Records;  
}

package com.example.myapplication.models.viewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class EmployeeViewModel {  
 @SerializedName("id")  
 @Expose  
 private Integer Id;  
  
 @SerializedName("fio")  
 @Expose  
 private String FIO;  
  
 public EmployeeViewModel(String FIO, String login, String password) {  
 this.FIO = FIO;  
 Login = login;  
 Password = password;  
 }  
  
 @SerializedName("login")  
 @Expose  
 private String Login;  
  
 @SerializedName("password")  
 @Expose  
 private String Password;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public String getFIO() {  
 return FIO;  
 }  
  
 public void setFIO(String FIO) {  
 this.FIO = FIO;  
 }  
  
 public String getLogin() {  
 return Login;  
 }  
  
 public void setLogin(String login) {  
 Login = login;  
 }  
  
 public String getPassword() {  
 return Password;  
 }  
  
 public void setPassword(String password) {  
 Password = password;  
 }  
}

package com.example.myapplication.models.viewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class ServiceRecordViewModel {  
 @SerializedName("id")  
 @Expose  
 private Integer Id;  
  
 @SerializedName("carId")  
 @Expose  
 private Integer CarId;  
  
 public ServiceRecordViewModel(Integer carId, String carBrandAndName,  
 String dateBegin, String dateEnd,  
 String description) {  
 CarId = carId;  
 CarBrandAndName = carBrandAndName;  
 DateBegin = dateBegin.split("T")[0] + " " + dateBegin.split("T")[1].substring(0, 5);  
 DateEnd = dateEnd.split("T")[0] + " " + dateEnd.split("T")[1].substring(0, 5);  
 Description = description;  
 }  
  
 @SerializedName("carBrandAndName")  
 @Expose  
 private String CarBrandAndName;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public Integer getCarId() {  
 return CarId;  
 }  
  
 public void setCarId(Integer carId) {  
 CarId = carId;  
 }  
  
 public String getCarBrandAndName() {  
 return CarBrandAndName;  
 }  
  
 public void setCarBrandAndName(String carBrandAndName) {  
 CarBrandAndName = carBrandAndName;  
 }  
  
 public String getDateBegin() {  
 return DateBegin;  
 }  
  
 public void setDateBegin(String dateBegin) {  
 DateBegin = dateBegin.split("T")[0] + " " + dateBegin.split("T")[1].substring(0, 5);  
 }  
  
 public String getDateEnd() {  
 return DateEnd;  
 }  
  
 public void setDateEnd(String dateEnd) {  
 DateEnd = dateEnd.split("T")[0] + " " + dateEnd.split("T")[1].substring(0, 5);  
 }  
  
 public String getDescription() {  
 return Description;  
 }  
  
 public void setDescription(String description) {  
 Description = description;  
 }  
  
 @SerializedName("dateBegin")  
 @Expose  
 private String DateBegin;  
  
 @SerializedName("dateEnd")  
 @Expose  
 private String DateEnd;  
  
 @SerializedName("description")  
 @Expose  
 private String Description;  
}

package com.example.myapplication.models.viewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class SparePartViewModel {  
 @SerializedName("id")  
 @Expose  
 private Integer Id;  
  
 public SparePartViewModel(String name, String factoryNumber, Float price, String type, String UMeasurement) {  
 Name = name;  
 FactoryNumber = factoryNumber;  
 Price = price;  
 Type = type;  
 this.UMeasurement = UMeasurement;  
 }  
  
 @SerializedName("name")  
 @Expose  
 private String Name;  
  
 @SerializedName("factoryNumber")  
 @Expose  
 private String FactoryNumber;  
  
 @SerializedName("price")  
 @Expose  
 private Float Price;  
  
 @SerializedName("type")  
 @Expose  
 private String Type;  
  
 @SerializedName("uMeasurement")  
 @Expose  
 private String UMeasurement;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public String getName() {  
 return Name;  
 }  
  
 public void setName(String name) {  
 Name = name;  
 }  
  
 public String getFactoryNumber() {  
 return FactoryNumber;  
 }  
  
 public void setFactoryNumber(String factoryNumber) {  
 FactoryNumber = factoryNumber;  
 }  
  
 public Float getPrice() {  
 return Price;  
 }  
  
 public void setPrice(Float price) {  
 Price = price;  
 }  
  
 public String getType() {  
 return Type;  
 }  
  
 public void setType(String type) {  
 Type = type;  
 }  
  
 public String getUMeasurement() {  
 return UMeasurement;  
 }  
  
 public void setUMeasurement(String UMeasurement) {  
 this.UMeasurement = UMeasurement;  
 }  
}

package com.example.myapplication.models.viewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class StoreKeeperViewModel {  
 @SerializedName("id")  
 @Expose  
 private Integer Id;  
  
 @SerializedName("fio")  
 @Expose  
 private String FIO;  
  
 public StoreKeeperViewModel(String FIO, String login,  
 String password) {  
 this.FIO = FIO;  
 Login = login;  
 Password = password;  
 }  
  
 @SerializedName("login")  
 @Expose  
 private String Login;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public String getFIO() {  
 return FIO;  
 }  
  
 public void setFIO(String FIO) {  
 this.FIO = FIO;  
 }  
  
 public String getLogin() {  
 return Login;  
 }  
  
 public void setLogin(String login) {  
 Login = login;  
 }  
  
 public String getPassword() {  
 return Password;  
 }  
  
 public void setPassword(String password) {  
 Password = password;  
 }  
  
 @SerializedName("password")  
 @Expose  
 private String Password;  
}

package com.example.myapplication.models.viewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
import java.util.Map;  
  
public class TOViewModel {  
 @SerializedName("id")  
 @Expose  
 private Integer Id;  
  
 @SerializedName("carId")  
 @Expose  
 private Integer CarId;  
  
 @SerializedName("carName")  
 @Expose  
 private String CarName;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public Integer getCarId() {  
 return CarId;  
 }  
  
 public void setCarId(Integer carId) {  
 CarId = carId;  
 }  
  
 public String getCarName() {  
 return CarName;  
 }  
  
 public void setCarName(String carName) {  
 CarName = carName;  
 }  
  
 public Integer getEmployeeId() {  
 return EmployeeId;  
 }  
  
 public void setEmployeeId(Integer employeeId) {  
 EmployeeId = employeeId;  
 }  
  
 public String getEmployeeName() {  
 return EmployeeName;  
 }  
  
 public void setEmployeeName(String employeeName) {  
 EmployeeName = employeeName;  
 }  
  
 public String getTOAndCarName() {  
 return TOAndCarName;  
 }  
  
 public void setTOAndCarName(String TOAndCarName) {  
 this.TOAndCarName = TOAndCarName;  
 }  
  
 public Float getSum() {  
 return Sum;  
 }  
  
 public void setSum(Float sum) {  
 Sum = sum;  
 }  
  
 public String getStatus() {  
 return Status;  
 }  
  
 public void setStatus(String status) {  
 Status = status;  
 }  
  
 public String getDateCreate() {  
 return DateCreate;  
 }  
  
 public void setDateCreate(String dateCreate) {  
 DateCreate = dateCreate;  
 }  
  
 public String getDateImplement() {  
 if (DateImplement == null)  
 return "-";  
 return DateImplement;  
 }  
  
 public void setDateImplement(String dateImplement) {  
 DateImplement = dateImplement;  
 }  
  
 public String getDateOver() {  
 if (DateOver == null)  
 return "-";  
 return DateOver;  
 }  
  
 public void setDateOver(String dateOver) {  
 DateOver = dateOver;  
 }  
  
 public Map<Integer, String> getWorks() {  
 return Works;  
 }  
  
 public void setWorks(Map<Integer, String> works) {  
 Works = works;  
 }  
  
 @SerializedName("employeeId")  
 @Expose  
 private Integer EmployeeId;  
  
 public TOViewModel(Integer carId, String carName,  
 Integer employeeId, String employeeName,  
 String TOAndCarName, Float sum,  
 String status, String dateCreate,  
 String dateImplement, String dateOver,  
 Map<Integer, String> works) {  
 CarId = carId;  
 CarName = carName;  
 EmployeeId = employeeId;  
 EmployeeName = employeeName;  
 this.TOAndCarName = TOAndCarName;  
 Sum = sum;  
 Status = status;  
 DateCreate = dateCreate;  
 DateImplement = dateImplement;  
 DateOver = dateOver;  
 Works = works;  
 }  
  
 @SerializedName("employeeName")  
 @Expose  
 private String EmployeeName;  
  
 @SerializedName("toAndCarName")  
 @Expose  
 private String TOAndCarName;  
  
 @SerializedName("sum")  
 @Expose  
 private Float Sum;  
  
 @SerializedName("status")  
 @Expose  
 private String Status;  
  
 @SerializedName("dateCreate")  
 @Expose  
 private String DateCreate;  
  
 @SerializedName("dateImplement")  
 @Expose  
 private String DateImplement;  
  
 @SerializedName("dateOver")  
 @Expose  
 private String DateOver;  
  
 /// <summary>  
 /// Работы ТО (номер, (название, (кол-во, стоимость  
 /// </summary>  
 @SerializedName("works")  
 @Expose  
 private Map<Integer, String> Works;  
}

package com.example.myapplication.models.viewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
import java.util.Map;  
  
public class WorkTypeViewModel {  
 @SerializedName("id")  
 @Expose  
 public Integer Id;  
  
 @SerializedName("timeOfWorkId")  
 @Expose  
 public Integer TimeOfWorkId;  
  
 @SerializedName("executionTime")  
 @Expose  
 public String ExecutionTime;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public Integer getTimeOfWorkId() {  
 return TimeOfWorkId;  
 }  
  
 public void setTimeOfWorkId(Integer timeOfWorkId) {  
 TimeOfWorkId = timeOfWorkId;  
 }  
  
 public String getExecutionTime() {  
 return ExecutionTime;  
 }  
  
 public void setExecutionTime(String executionTime) {  
 ExecutionTime = executionTime;  
 }  
  
 public String getWorkName() {  
 return WorkName;  
 }  
  
 public void setWorkName(String workName) {  
 WorkName = workName;  
 }  
  
 public Float getPrice() {  
 return Price;  
 }  
  
 public void setPrice(Float price) {  
 Price = price;  
 }  
  
 public Float getNetPrice() {  
 return NetPrice;  
 }  
  
 public void setNetPrice(Float netPrice) {  
 NetPrice = netPrice;  
 }  
  
 public Map<Integer, Float> getWorkSpareParts() {  
 return WorkSpareParts;  
 }  
  
 public void setWorkSpareParts(Map<Integer, Float> workSpareParts) {  
 WorkSpareParts = workSpareParts;  
 }  
  
 public WorkTypeViewModel(Integer timeOfWorkId, String executionTime, String workName,  
 Float price, Float netPrice, Map<Integer,  
 Float> workSpareParts) {  
 TimeOfWorkId = timeOfWorkId;  
 ExecutionTime = executionTime;  
 WorkName = workName;  
 Price = price;  
 NetPrice = netPrice;  
 WorkSpareParts = workSpareParts;  
 }  
  
 @SerializedName("workName")  
 @Expose  
 public String WorkName;  
  
 @SerializedName("price")  
 @Expose  
 public Float Price;  
  
 @SerializedName("netPrice")  
 @Expose  
 public Float NetPrice;  
  
 /// <summary>  
 /// Необходимые детали и расходники (int - id, string - название, decimal, потому что может быть не целое (например, 0.8 л масла))  
 /// </summary>  
 @SerializedName("workSpareParts")  
 @Expose  
 public Map<Integer, Float> WorkSpareParts;  
}

package com.example.myapplication.models.viewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class WorkViewModel {  
  
 @SerializedName("id")  
 @Expose  
 public Integer Id;  
  
 @SerializedName("storeKeeperId")  
 @Expose  
 public Integer StoreKeeperId;  
  
 @SerializedName("storeKeeperFIO")  
 @Expose  
 public String StoreKeeperFIO;  
  
 @SerializedName("workTypeId")  
 @Expose  
 public Integer WorkTypeId;  
  
 @SerializedName("toId")  
 @Expose  
 public Integer TOId;  
  
 @SerializedName("workName")  
 @Expose  
 public String WorkName;  
  
 public Integer getId() {  
 return Id;  
 }  
  
 public void setId(Integer id) {  
 Id = id;  
 }  
  
 public Integer getStoreKeeperId() {  
 return StoreKeeperId;  
 }  
  
 public void setStoreKeeperId(Integer storeKeeperId) {  
 StoreKeeperId = storeKeeperId;  
 }  
  
 public String getStoreKeeperFIO() {  
 return StoreKeeperFIO;  
 }  
  
 public void setStoreKeeperFIO(String storeKeeperFIO) {  
 StoreKeeperFIO = storeKeeperFIO;  
 }  
  
 public Integer getWorkTypeId() {  
 return WorkTypeId;  
 }  
  
 public void setWorkTypeId(Integer workTypeId) {  
 WorkTypeId = workTypeId;  
 }  
  
 public Integer getTOId() {  
 return TOId;  
 }  
  
 public void setTOId(Integer TOId) {  
 this.TOId = TOId;  
 }  
  
 public String getWorkName() {  
 return WorkName;  
 }  
  
 public void setWorkName(String workName) {  
 WorkName = workName;  
 }  
  
 public Float getPrice() {  
 return Price;  
 }  
  
 public void setPrice(Float price) {  
 Price = price;  
 }  
  
 public Float getNetPrice() {  
 return NetPrice;  
 }  
  
 public void setNetPrice(Float netPrice) {  
 NetPrice = netPrice;  
 }  
  
 public String getWorkStatus() {  
 return WorkStatus;  
 }  
  
 public void setWorkStatus(String workStatus) {  
 WorkStatus = workStatus;  
 }  
  
 public String getWorkBegin() {  
 if (WorkBegin == null)  
 return "-";  
 return WorkBegin;  
 }  
  
 public void setWorkBegin(String workBegin) {  
 WorkBegin = workBegin;  
 }  
  
 public Integer getCount() {  
 return Count;  
 }  
  
 public void setCount(Integer count) {  
 Count = count;  
 }  
  
 @SerializedName("price")  
 @Expose  
 public Float Price;  
  
 @SerializedName("netPrice")  
 @Expose  
 public Float NetPrice;  
  
 public WorkViewModel(Integer storeKeeperId, String storeKeeperFIO, Integer workTypeId,  
 Integer TOId, String workName, Float price,  
 Float netPrice, String workStatus,  
 String workBegin, Integer count) {  
 StoreKeeperId = storeKeeperId;  
 StoreKeeperFIO = storeKeeperFIO;  
 WorkTypeId = workTypeId;  
 this.TOId = TOId;  
 WorkName = workName;  
 Price = price;  
 NetPrice = netPrice;  
 WorkStatus = workStatus;  
 WorkBegin = workBegin;  
 Count = count;  
 }  
  
 @SerializedName("workStatus")  
 @Expose  
 public String WorkStatus;  
  
 @SerializedName("workBegin")  
 @Expose  
 public String WorkBegin;  
  
 @SerializedName("count")  
 @Expose  
 public Integer Count;  
}

package com.example.myapplication.models.webViewModels;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class SparePartWebViewModel {  
  
 public SparePartWebViewModel(Float count, String name, String factoryNumber, Float price, String type, String UMeasurement) {  
 Count = count;  
 Name = name;  
 FactoryNumber = factoryNumber;  
 Price = price;  
 Type = type;  
 this.UMeasurement = UMeasurement;  
 }  
  
 @SerializedName("count")  
 @Expose  
 private Float Count;  
  
 @SerializedName("name")  
 @Expose  
 private String Name;  
  
 @SerializedName("factoryNumber")  
 @Expose  
 private String FactoryNumber;  
  
 @SerializedName("price")  
 @Expose  
 private Float Price;  
  
 @SerializedName("type")  
 @Expose  
 private String Type;  
  
 @SerializedName("uMeasurement")  
 @Expose  
 private String UMeasurement;  
  
 public Float getCount() {  
 return Count;  
 }  
  
 public void setCount(Float count) {  
 Count = count;  
 }  
  
 public String getName() {  
 return Name;  
 }  
  
 public void setName(String name) {  
 Name = name;  
 }  
  
 public String getFactoryNumber() {  
 return FactoryNumber;  
 }  
  
 public void setFactoryNumber(String factoryNumber) {  
 FactoryNumber = factoryNumber;  
 }  
  
 public Float getPrice() {  
 return Price;  
 }  
  
 public void setPrice(Float price) {  
 Price = price;  
 }  
  
 public String getType() {  
 return Type;  
 }  
  
 public void setType(String type) {  
 Type = type;  
 }  
  
 public String getUMeasurement() {  
 return UMeasurement;  
 }  
  
 public void setUMeasurement(String UMeasurement) {  
 this.UMeasurement = UMeasurement;  
 }  
}

package com.example.myapplication.service;  
  
import com.example.myapplication.models.bindingModels.CarBindingModel;  
import com.example.myapplication.models.bindingModels.ChangeTOStatusBindingModel;  
import com.example.myapplication.models.bindingModels.CreateTOBindingModel;  
import com.example.myapplication.models.bindingModels.CreateWorkBindingModel;  
import com.example.myapplication.models.bindingModels.EmployeeBindingModel;  
import com.example.myapplication.models.bindingModels.ServiceRecordBindingModel;  
import com.example.myapplication.models.viewModels.CarViewModel;  
import com.example.myapplication.models.viewModels.EmployeeViewModel;  
import com.example.myapplication.models.viewModels.ServiceRecordViewModel;  
import com.example.myapplication.models.viewModels.StoreKeeperViewModel;  
import com.example.myapplication.models.viewModels.TOViewModel;  
import com.example.myapplication.models.viewModels.WorkTypeViewModel;  
import com.example.myapplication.models.viewModels.WorkViewModel;  
import com.example.myapplication.models.webViewModels.SparePartWebViewModel;  
  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.http.Body;  
import retrofit2.http.GET;  
import retrofit2.http.POST;  
import retrofit2.http.Query;  
  
public interface STOApi {  
  
 //Авто  
 @GET("Car/CarList")  
 Call<List<CarViewModel>> getCars();  
  
 @GET("Car/GetCar")  
 Call<CarViewModel> getCar(@Query("carId") int carId);  
  
 @GET("Car/GetServiceRecords")  
 Call<List<ServiceRecordViewModel>> getServiceRecords(@Query("carId") int carId);  
  
 @POST("Car/CreateOrUpdateCar")  
 Call<Void> createOrUpdateCar(@Body CarBindingModel car);  
  
 @POST("Car/DeleteCar")  
 Call<Void> deleteCar(@Body CarBindingModel car);  
  
 @GET("Car/GetServiceRecord")  
 Call<ServiceRecordViewModel> getServiceRecord(@Query("servicerecordId") int recordId);  
  
 @POST("Car/UpdateServiceRecord")  
 Call<Void> updateRecord(@Body ServiceRecordBindingModel record);  
  
 //TO  
 @GET("TO/GetTOList")  
 Call<List<TOViewModel>> getTOs(@Query("employeeId") Integer employeeId);  
  
 @GET("TO/GetNotStartedTOs")  
 Call<List<TOViewModel>> getNotStartedTOs(@Query("employeeId") Integer employeeId);  
  
 @GET("TO/GetTO")  
 Call<TOViewModel> getTO(@Query("tOId") Integer toId);  
  
 @POST("TO/CreateTO")  
 Call<Void> createTO(@Body CreateTOBindingModel to);  
  
 @POST("TO/TakeTOInWork")  
 Call<Boolean> takeTOInWork(@Body ChangeTOStatusBindingModel to);  
  
 @POST("TO/FinishTO")  
 Call<Boolean> finishTO(@Body ChangeTOStatusBindingModel to);  
  
 @POST("TO/IssueTO")  
 Call<Boolean> issueTO(@Body ChangeTOStatusBindingModel to);  
  
 //WorkType  
 @GET("Work/GetWorkTypeList")  
 Call<List<WorkTypeViewModel>> getWorkTypes();  
  
 @GET("Work/GetWorkType")  
 Call<WorkTypeViewModel> getWorkType(@Query("workTypeId") int workTypeId);  
  
 //Work  
 @GET("Work/GetWorkListByEmployee")  
 Call<List<WorkViewModel>> getWorksByEmployee(@Query("employeeId") int employeeId);  
  
 @POST("Work/CreateWork")  
 Call<Void> createWork(@Body CreateWorkBindingModel work);  
  
 @GET("Work/GetWorkListByTO")  
 Call<List<WorkViewModel>> getWorksByTO(@Query("toId") Integer toId);  
  
 @GET("Work/GetWork")  
 Call<WorkViewModel> getWork(@Query("workId") Integer workId);  
  
 //StoreKeeper  
 @GET("StoreKeeper/GetStoreKeeperList")  
 Call<List<StoreKeeperViewModel>> getStoreKeepers();  
  
 //Employee  
 @GET("Employee/Login")  
 Call<EmployeeViewModel> logIn(@Query("login") String login, @Query("password") String password);  
  
 @POST("Employee/Register")  
 Call<Void> singUp(@Body EmployeeBindingModel model);  
  
 @POST("Employee/UpdateData")  
 Call<Void> updateProfile(@Body EmployeeBindingModel model);  
  
 @POST("Employee/Delete")  
 Call<Void> deleteProfile(@Body EmployeeBindingModel model);  
  
 //SparePart  
 @GET("SparePart/GetSparePartsByWorkType")  
 Call<List<SparePartWebViewModel>> getSparePartsByWorkType(@Query("workTypeId") Integer workTypeId);  
}

package com.example.myapplication.service;  
  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
  
import org.jetbrains.annotations.NotNull;  
  
import java.io.IOException;  
  
import okhttp3.HttpUrl;  
import okhttp3.Interceptor;  
import okhttp3.OkHttpClient;  
import okhttp3.Request;  
import okhttp3.Response;  
import okhttp3.logging.HttpLoggingInterceptor;  
import retrofit2.Retrofit;  
import retrofit2.adapter.rxjava2.RxJava2CallAdapterFactory;  
import retrofit2.converter.gson.GsonConverterFactory;  
  
public class STOService {  
 private final STOApi api;  
  
 private final String URL = "http://192.168.0.187:7252/api/";  
 public STOService() {  
 Retrofit mRetrofit = createRetrofit();  
 api = mRetrofit.create(STOApi.class);  
 }  
  
 public STOApi getApi() {  
 return api;  
 }  
  
 private OkHttpClient createOkHttpClient() {  
 final OkHttpClient.Builder httpClient = new OkHttpClient.Builder();  
 httpClient.addInterceptor(new Interceptor() {  
 @NotNull  
 @Override  
 public Response intercept(@NotNull Interceptor.Chain chain) throws IOException {  
 final Request original = chain.request();  
 final HttpUrl originalHttpUrl = original.url();  
 final HttpUrl url = originalHttpUrl.newBuilder()  
 .build();  
 final Request.Builder requestBuilder = original.newBuilder()  
 .url(url);  
 final Request request = requestBuilder.build();  
 return chain.proceed(request);  
 }  
 });  
 HttpLoggingInterceptor logging = new HttpLoggingInterceptor();  
 logging.level(HttpLoggingInterceptor.Level.BODY);  
 httpClient.addInterceptor(logging);  
  
 return httpClient.build();  
 }  
  
 private Retrofit createRetrofit() {  
 return new Retrofit.Builder()  
 .baseUrl(URL)  
 .addConverterFactory(GsonConverterFactory.create(getGSON()))  
 .client(createOkHttpClient())  
 .addCallAdapterFactory(RxJava2CallAdapterFactory.create())  
 .build();  
 }  
  
 private Gson getGSON() {  
 return new GsonBuilder()  
 .serializeNulls()  
 .enableComplexMapKeySerialization().create();  
 }  
}

package com.example.myapplication.spinners;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.CarViewModel;  
import com.example.myapplication.models.viewModels.TOViewModel;  
  
import java.util.ArrayList;  
  
public class CarSpinnerAdapter extends ArrayAdapter<CarViewModel> {  
  
 private ArrayList<CarViewModel> cars;  
 private final Context context;  
  
 public CarSpinnerAdapter(@NonNull Context context) {  
 super(context, 0);  
 this.context = context;  
 cars = new ArrayList<>();  
 }  
  
 public void setData(ArrayList<CarViewModel> objects) {  
 cars = objects;  
 }  
  
 @Override  
 public int getCount() {  
 return cars.size();  
 }  
  
 @Override  
 public CarViewModel getItem(int i) {  
 return cars.get(i);  
 }  
  
 @Override  
 public long getItemId(int i) {  
 return cars.get(i).getId();  
 }  
  
 @Override  
 public View getView(int i, View view, ViewGroup viewGroup) {  
 return initView(i, view, viewGroup);  
 }  
  
 // It gets a View that displays in the drop down popup the data at the specified position  
 @Override  
 public View getDropDownView(int position, View convertView,  
 @NonNull ViewGroup parent) {  
 return initView(position, convertView, parent);  
 }  
  
 private View initView(int position, View convertView,  
 ViewGroup parent) {  
 if (cars.size() > 0) {  
 if (convertView == null) {  
 convertView = LayoutInflater.from(context).inflate(R.layout.spinner\_cars\_item\_closed, parent, false);  
 }  
 CarViewModel car = cars.get(position);  
 TextView textView = convertView.findViewById(R.id.spinner\_car\_item\_car);  
 String name = car.getBrand() + " " + car.getModel();  
 if (name.length() > 8)  
 textView.setText(name.substring(0, 8) + '.');  
 else  
 textView.setText(name);  
 TextView textView1 = convertView.findViewById(R.id.spinner\_car\_item\_vin);  
 textView1.setText("VIN:\n" + car.getVIN());  
 }  
 return convertView;  
 }  
}

package com.example.myapplication.spinners;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.StoreKeeperViewModel;  
import com.example.myapplication.models.viewModels.WorkTypeViewModel;  
  
import java.util.ArrayList;  
  
public class StoreKeeperSpinnerAdapter extends ArrayAdapter<StoreKeeperViewModel> {  
 private ArrayList<StoreKeeperViewModel> storeKeepers;  
 private final Context context;  
  
 public StoreKeeperSpinnerAdapter(@NonNull Context context) {  
 super(context, 0);  
 this.context = context;  
 storeKeepers = new ArrayList<>();  
 }  
  
 @Override  
 public int getCount() {  
 return storeKeepers.size();  
 }  
  
 public void setData(ArrayList<StoreKeeperViewModel> objects) {  
 storeKeepers = objects;  
 }  
  
 @Override  
 public StoreKeeperViewModel getItem(int i) {  
 return storeKeepers.get(i);  
 }  
  
 @Override  
 public long getItemId(int i) {  
 return storeKeepers.get(i).getId();  
 }  
  
 @Override  
 public View getView(int i, View view, ViewGroup viewGroup) {  
 return initView(i, view, viewGroup);  
 }  
  
 // It gets a View that displays in the drop down popup the data at the specified position  
 @Override  
 public View getDropDownView(int position, View convertView,  
 @NonNull ViewGroup parent) {  
 return initView(position, convertView, parent);  
 }  
  
 private View initView(int position, View convertView,  
 ViewGroup parent) {  
 if (storeKeepers.size() > 0) {  
 if (convertView == null) {  
 convertView = LayoutInflater.from(context).inflate(R.layout.spinner\_storekeeper\_item\_closed, parent, false);  
 }  
  
 TextView textView = convertView.findViewById(R.id.spinner\_storekeeper\_item\_text);  
 textView.setText(storeKeepers.get(position).getFIO());  
 }  
 return convertView;  
 }  
}

package com.example.myapplication.spinners;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.StoreKeeperViewModel;  
import com.example.myapplication.models.viewModels.TOViewModel;  
  
import java.util.ArrayList;  
  
public class TOSpinnerAdapter extends ArrayAdapter<TOViewModel> {  
 private ArrayList<TOViewModel> toS;  
 private final Context context;  
  
 public TOSpinnerAdapter(@NonNull Context context) {  
 super(context, 0);  
 this.context = context;  
 toS = new ArrayList<>();  
 }  
  
 public void setData(ArrayList<TOViewModel> objects) {  
 toS = objects;  
 }  
  
 @Override  
 public int getCount() {  
 return toS.size();  
 }  
  
 @Override  
 public TOViewModel getItem(int i) {  
 return toS.get(i);  
 }  
  
 @Override  
 public long getItemId(int i) {  
 return toS.get(i).getId();  
 }  
  
 @Override  
 public View getView(int i, View view, ViewGroup viewGroup) {  
 return initView(i, view, viewGroup);  
 }  
  
 // It gets a View that displays in the drop down popup the data at the specified position  
 @Override  
 public View getDropDownView(int position, View convertView,  
 @NonNull ViewGroup parent) {  
 return initView(position, convertView, parent);  
 }  
  
 private View initView(int position, View convertView,  
 ViewGroup parent) {  
 if (toS.size() > 0) {  
 if (convertView == null) {  
 convertView = LayoutInflater.from(context).inflate(R.layout.spinner\_tos\_item\_closed, parent, false);  
 }  
 TOViewModel to = toS.get(position);  
 TextView textView = convertView.findViewById(R.id.spinner\_to\_item\_car\_and\_number);  
 String name = to.getTOAndCarName();  
 if (name.length() > 8)  
 textView.setText(name.substring(0, 8) + '.');  
 else  
 textView.setText(name);  
 TextView textView1 = convertView.findViewById(R.id.spinner\_to\_item\_datecreate);  
 textView1.setText("Дата создания: " + to.getDateCreate().split("T")[0] + " " + to.getDateCreate().split("T")[1].substring(0, 5));  
 }  
 return convertView;  
 }  
}

package com.example.myapplication.spinners;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.TOViewModel;  
import com.example.myapplication.models.viewModels.WorkTypeViewModel;  
  
import java.util.ArrayList;  
  
public class WorkTypeSpinnerAdapter extends ArrayAdapter<WorkTypeViewModel> {  
 private ArrayList<WorkTypeViewModel> workTypes;  
 private final Context context;  
  
 public WorkTypeSpinnerAdapter(@NonNull Context context) {  
 super(context, 0);  
 this.context = context;  
 workTypes = new ArrayList<>();  
 }  
  
 public void setData(@NonNull ArrayList<WorkTypeViewModel> objects) {  
 workTypes = objects;  
 }  
  
 @Override  
 public int getCount() {  
 return workTypes.size();  
 }  
  
 @Override  
 public WorkTypeViewModel getItem(int i) {  
 return workTypes.get(i);  
 }  
  
 @Override  
 public long getItemId(int i) {  
 return workTypes.get(i).getId();  
 }  
  
 @Override  
 public View getView(int i, View view, ViewGroup viewGroup) {  
 return initView(i, view, viewGroup);  
 }  
  
 // It gets a View that displays in the drop down popup the data at the specified position  
 @Override  
 public View getDropDownView(int position, View convertView,  
 @NonNull ViewGroup parent) {  
 return initView(position, convertView, parent);  
 }  
  
 private View initView(int position, View convertView,  
 ViewGroup parent) {  
 if (workTypes.size() > 0) {  
 if (convertView == null) {  
 convertView = LayoutInflater.from(context).inflate(R.layout.spinner\_worktypes\_item\_closed, parent, false);  
 }  
 WorkTypeViewModel work = workTypes.get(position);  
 TextView textView = convertView.findViewById(R.id.spinner\_worktype\_item\_name);  
 textView.setText(work.getWorkName());  
 TextView textView1 = convertView.findViewById(R.id.spinner\_worktype\_item\_netprice);  
 textView1.setText(String.valueOf(work.getNetPrice()));  
 }  
 return convertView;  
 }  
}

package com.example.myapplication.ui.cars;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentTransaction;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.text.TextUtils;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.adapters.RecordListAdapter;  
import com.example.myapplication.models.bindingModels.CarBindingModel;  
import com.example.myapplication.models.viewModels.CarViewModel;  
import com.example.myapplication.models.viewModels.ServiceRecordViewModel;  
import com.example.myapplication.ui.records.EditRecordFragment;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
  
public class AddEditCarFragment extends Fragment {  
 private RecordListAdapter adapter;  
 private RecyclerView recyclerView;  
 private ArrayList<ServiceRecordViewModel> records;  
 private App app;  
 private Integer carId;  
 private CarViewModel car;  
 private EditText editText\_brand;  
 private EditText editText\_model;  
 private EditText editText\_vin;  
 private EditText editText\_phone\_num;  
 private Button button\_car\_edit;  
 public AddEditCarFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.car\_add\_edit\_fragment, container, false);  
 recyclerView = view.findViewById(R.id.recyclerview\_records);  
  
 button\_car\_edit = view.findViewById(R.id.button\_car\_edit);  
  
 editText\_brand = view.findViewById(R.id.watch\_work\_storekeeperfioTV);  
 editText\_model = view.findViewById(R.id.create\_work\_priceET);  
 editText\_vin = view.findViewById(R.id.editText\_vin);  
 editText\_phone\_num = view.findViewById(R.id.editText\_phone\_number);  
  
 setRecyclerView();  
  
 button\_car\_edit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 createOrUpdate();  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 getActivity().getSupportFragmentManager()  
 .popBackStack();  
 }  
 });  
 // Inflate the layout for this fragment  
 return view;  
 }  
  
 private void setRecyclerView() {  
 Bundle bundle = getArguments();  
 if (bundle != null) {  
 carId = bundle.getInt("carId");  
 button\_car\_edit.setText("Изменить");  
 recyclerView.setHasFixedSize(true);  
 recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));  
  
 adapter = new RecordListAdapter(getContext());  
 recyclerView.setAdapter(adapter);  
 restore();  
 //при долгом нажатии на элемент открывается фрагмент его редактирования  
 adapter.setOnItemLongClickListener(new RecordListAdapter.OnItemLongClickListener() {  
 @Override  
 public void onItemLongClicked(ServiceRecordViewModel record, int position) {  
 Bundle bundle = new Bundle();  
 Integer id = record.getId();  
 bundle.putInt("recordId", id);  
 EditRecordFragment editRecordFragment = new EditRecordFragment();  
 editRecordFragment.setArguments(bundle);  
 replaceFragment(editRecordFragment);  
 }  
 });  
 }  
 }  
  
 private void restore() {  
 if (carId != null){  
 Call<CarViewModel> call = app.getStoService().getApi().getCar(carId);  
 call.enqueue(new Callback<CarViewModel>() {  
 @Override  
 public void onResponse(@NonNull Call<CarViewModel> call, @NonNull Response<CarViewModel> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 car = response.body();  
 editText\_brand.setText(car.getBrand());  
 editText\_model.setText(car.getModel());  
 editText\_vin.setText(car.getVIN());  
 editText\_phone\_num.setText(car.getOwnerPhoneNumber());  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<CarViewModel> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
  
 Call<List<ServiceRecordViewModel>> call2 = app.getStoService().getApi().getServiceRecords(carId);  
 call2.enqueue(new Callback<List<ServiceRecordViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<ServiceRecordViewModel>> call, @NonNull Response<List<ServiceRecordViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 records = (ArrayList<ServiceRecordViewModel>) response.body();  
 adapter.setData(records);  
 adapter.notifyDataSetChanged();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<ServiceRecordViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
 }  
  
 private void createOrUpdate() {  
 if (!TextUtils.isEmpty(editText\_brand.getText()) &&  
 !TextUtils.isEmpty(editText\_model.getText()) &&  
 !TextUtils.isEmpty(editText\_vin.getText()) &&  
 !TextUtils.isEmpty(editText\_phone\_num.getText())) {  
  
  
 String brand = String.valueOf(editText\_brand.getText());  
 String model = String.valueOf(editText\_model.getText());  
 String vin = String.valueOf(editText\_vin.getText());  
 String phone = String.valueOf(editText\_phone\_num.getText());  
  
 CarBindingModel bind\_car = new CarBindingModel();  
  
 bind\_car.setBrand(brand);  
 bind\_car.setModel(model);  
 bind\_car.setVIN(vin);  
 bind\_car.setOwnerPhoneNumber(phone);  
  
 if (car != null) {  
  
 bind\_car.setId(car.getId());  
 bind\_car.setRecords(car.getRecords());  
  
 createOrUpdateCar(bind\_car);  
 } else {  
 createOrUpdateCar(bind\_car);  
 }  
 }  
  
 }  
  
 private void createOrUpdateCar(CarBindingModel bind\_car) {  
 Call<Void> call = app.getStoService().getApi().createOrUpdateCar(bind\_car);  
 call.enqueue(new Callback<Void>() {  
 @Override  
 public void onResponse(@NonNull Call<Void> call, @NonNull Response<Void> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Void> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void replaceFragment(Fragment fragment) {  
 FragmentTransaction fragmentTransaction = getActivity()  
 .getSupportFragmentManager()  
 .beginTransaction();  
  
 fragmentTransaction.replace(R.id.frame\_layout, fragment);  
 fragmentTransaction.addToBackStack("");  
 fragmentTransaction.commit();  
 }  
}

package com.example.myapplication.ui.cars;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentTransaction;  
import androidx.recyclerview.widget.ItemTouchHelper;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.adapters.CarListAdapter;  
import com.example.myapplication.models.bindingModels.CarBindingModel;  
import com.example.myapplication.models.viewModels.CarViewModel;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class CarsFragment extends Fragment {  
 private CarListAdapter adapter;  
 private RecyclerView recyclerView;  
 private ArrayList<CarViewModel> cars;  
 private App app;  
 private FloatingActionButton fab;  
 public CarsFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.cars\_fragment, container, false);  
 recyclerView = view.findViewById(R.id.recyclerview\_cars);  
  
 fab = view.findViewById(R.id.fab\_add\_car);  
  
 fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 AddEditCarFragment addEditCarFragment = new AddEditCarFragment();  
 replaceFragment(addEditCarFragment);  
 }  
 });  
  
 setRecyclerView();  
 // Inflate the layout for this fragment  
 return view;  
 }  
  
 private void setRecyclerView() {  
 recyclerView.setHasFixedSize(true);  
 recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));  
  
 adapter = new CarListAdapter(getContext());  
 recyclerView.setAdapter(adapter);  
 restore();  
  
 //при долгом нажатии на элемент открывается фрагмент его редактирования  
 adapter.setOnItemLongClickListener(new CarListAdapter.OnItemLongClickListener() {  
 @Override  
 public void onItemLongClicked(CarViewModel car, int position) {  
 Bundle bundle = new Bundle();  
 Integer id = car.getId();  
 bundle.putInt("carId", id);  
 AddEditCarFragment addEditCarFragment = new AddEditCarFragment();  
 addEditCarFragment.setArguments(bundle);  
 replaceFragment(addEditCarFragment);  
 }  
 });  
  
 new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0,  
 ItemTouchHelper.RIGHT) {  
 @Override  
 public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder viewHolder1) {  
 return false;  
 }  
  
 @Override  
 public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int i) {  
 CarViewModel car = adapter.getCarAt(viewHolder.getAdapterPosition());  
 CarBindingModel car\_bind = new CarBindingModel();  
 car\_bind.setId(car.getId());  
 car\_bind.setBrand(car.getBrand());  
 car\_bind.setModel(car.getModel());  
 car\_bind.setVIN(car.getVIN());  
 car\_bind.setOwnerPhoneNumber(car.getOwnerPhoneNumber());  
 car\_bind.setRecords(car.getRecords());  
 deleteCar(car\_bind);  
  
 Toast.makeText(getContext(), "Авто удалено", Toast.LENGTH\_SHORT).show();  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 restore();  
 }  
 }).attachToRecyclerView(recyclerView);  
 }  
  
 private void restore() {  
 Call<List<CarViewModel>> call = app.getStoService().getApi().getCars();  
 call.enqueue(new Callback<List<CarViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<CarViewModel>> call, @NonNull Response<List<CarViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 cars = (ArrayList<CarViewModel>) response.body();  
 adapter.setData(cars);  
 adapter.notifyDataSetChanged();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<CarViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void deleteCar(CarBindingModel car) {  
 Call<Void> call = app.getStoService().getApi().deleteCar(car);  
 call.enqueue(new Callback<Void>() {  
 @Override  
 public void onResponse(@NonNull Call<Void> call, @NonNull Response<Void> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Void> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void replaceFragment(Fragment fragment) {  
 FragmentTransaction fragmentTransaction = getActivity()  
 .getSupportFragmentManager()  
 .beginTransaction();  
  
 fragmentTransaction.replace(R.id.frame\_layout, fragment);  
 fragmentTransaction.addToBackStack("");  
 fragmentTransaction.commit();  
 }  
}

package com.example.myapplication.ui.logregfragments;  
  
import android.content.Intent;  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentManager;  
import androidx.fragment.app.FragmentTransaction;  
  
import android.text.TextUtils;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.MainActivity;  
import com.example.myapplication.R;  
import com.example.myapplication.models.viewModels.EmployeeViewModel;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class LogInFragment extends Fragment {  
  
 private EditText et\_email;  
 private EditText et\_password;  
 private Button btn\_login;  
 private App app;  
 private ImageView swipe\_to\_register;  
 private Boolean flag;  
  
 public LogInFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_log\_in, container, false);  
  
 et\_email = view.findViewById(R.id.et\_email);  
 et\_password = view.findViewById(R.id.et\_password);  
 btn\_login = view.findViewById(R.id.btn\_login);  
 swipe\_to\_register = view.findViewById(R.id.swipe\_to\_register);  
  
 btn\_login.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 signIn();  
 }  
 });  
  
 swipe\_to\_register.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 RegisterFragment registerFragment = new RegisterFragment();  
 replaceFragment(registerFragment);  
 }  
 });  
 return view;  
 }  
  
 private void replaceFragment(Fragment fragment) {  
 FragmentManager fragmentManager = getActivity().getSupportFragmentManager();  
 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();  
  
 fragmentTransaction.replace(R.id.frame\_layout\_entry, fragment);  
 fragmentTransaction.commit();  
 }  
  
 private void signIn() {  
 if (!TextUtils.isEmpty(et\_email.getText()) && !TextUtils.isEmpty(et\_password.getText())) {  
 String mail = String.valueOf(et\_email.getText());  
 String password = String.valueOf(et\_password.getText());  
 Call<EmployeeViewModel> call = app.getStoService().getApi().logIn(mail, password);  
 call.enqueue(new Callback<EmployeeViewModel>() {  
 @Override  
 public void onResponse(@NonNull Call<EmployeeViewModel> call, @NonNull Response<EmployeeViewModel> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 App.setEmployee(response.body());  
 if (App.getEmployee() != null) {  
 Intent intent = new Intent(getActivity(), MainActivity.class);  
 startActivity(intent);  
 getActivity().finish();  
 }  
 else  
 Toast.makeText(getContext(), "Неверный пароль или логин!", Toast.LENGTH\_SHORT).show();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<EmployeeViewModel> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
 else {  
 Toast.makeText(getContext(), "Введите данные!", Toast.LENGTH\_SHORT).show();  
 }  
 }  
}

package com.example.myapplication.ui.logregfragments;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentManager;  
import androidx.fragment.app.FragmentTransaction;  
  
import android.text.TextUtils;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.models.bindingModels.EmployeeBindingModel;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class RegisterFragment extends Fragment {  
  
 private EditText et\_name;  
 private EditText et\_email;  
 private EditText et\_password;  
 private EditText et\_repassword;  
 private Button btn\_register;  
 private App app;  
 private ImageView swipe\_to\_login;  
 private Boolean flag;  
  
 public RegisterFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_register, container, false);  
 et\_name = view.findViewById(R.id.et\_name);  
 et\_email = view.findViewById(R.id.et\_email);  
 et\_password = view.findViewById(R.id.et\_password);  
 et\_repassword = view.findViewById(R.id.et\_repassword);  
 btn\_register = view.findViewById(R.id.btn\_register);  
 swipe\_to\_login = view.findViewById(R.id.swipe\_to\_login);  
  
 btn\_register.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 register();  
 }  
 });  
  
 swipe\_to\_login.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 LogInFragment logInFragment = new LogInFragment();  
 replaceFragment(logInFragment);  
 }  
 });  
 return view;  
 }  
  
 private void replaceFragment(Fragment fragment) {  
 FragmentManager fragmentManager = getActivity().getSupportFragmentManager();  
 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();  
  
 fragmentTransaction.replace(R.id.frame\_layout\_entry, fragment);  
 fragmentTransaction.commit();  
 }  
  
 private void register() {  
 if (!TextUtils.isEmpty(et\_email.getText()) && !TextUtils.isEmpty(et\_password.getText()) &&  
 !TextUtils.isEmpty(et\_repassword.getText()) && !TextUtils.isEmpty(et\_name.getText())) {  
  
 String password = String.valueOf(et\_password.getText());  
 String repassword = String.valueOf(et\_repassword.getText());  
  
 if (!password.equals(repassword)) {  
 Toast.makeText(getContext(), "Пароли должны совпадать!", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 String mail = String.valueOf(et\_email.getText());  
 String fio = String.valueOf(et\_name.getText());  
 EmployeeBindingModel model = new EmployeeBindingModel();  
 model.setFIO(fio);  
 model.setLogin(mail);  
 model.setPassword(password);  
 Call<Void> call = app.getStoService().getApi().singUp(model);  
 call.enqueue(new Callback<Void>() {  
 @Override  
 public void onResponse(@NonNull Call<Void> call, @NonNull Response<Void> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 LogInFragment logInFragment = new LogInFragment();  
 replaceFragment(logInFragment);  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Void> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
 else {  
 Toast.makeText(getContext(), "Введите данные!", Toast.LENGTH\_SHORT).show();  
 }  
 }  
}

package com.example.myapplication.ui.records;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
  
import android.text.TextUtils;  
import android.util.Log;  
import android.util.Pair;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.models.bindingModels.ServiceRecordBindingModel;  
import com.example.myapplication.models.viewModels.ServiceRecordViewModel;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
  
public class EditRecordFragment extends Fragment {  
 private ServiceRecordViewModel record;  
 private App app;  
 private Integer recordId;  
 private EditText record\_edit\_editText\_description;  
 private TextView record\_edit\_tv\_car\_brand\_and\_model;  
 private TextView record\_edit\_tv\_date\_begin;  
 private TextView record\_edit\_tv\_date\_end;  
 private Button record\_edit\_button\_edit\_record\_accept;  
  
 public EditRecordFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.record\_edit\_fragment, container, false);  
  
 //UI  
 record\_edit\_tv\_car\_brand\_and\_model = view.findViewById(R.id.record\_edit\_tv\_car\_brand\_and\_model);  
 record\_edit\_tv\_date\_begin = view.findViewById(R.id.record\_edit\_tv\_date\_begin);  
 record\_edit\_tv\_date\_end = view.findViewById(R.id.record\_edit\_tv\_date\_end);  
 record\_edit\_editText\_description = view.findViewById(R.id.record\_edit\_editText\_description);  
 record\_edit\_button\_edit\_record\_accept = view.findViewById(R.id.record\_edit\_button\_edit\_record\_accept);  
  
 loadData();  
  
 record\_edit\_button\_edit\_record\_accept.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 createOrUpdate();  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 getActivity().getSupportFragmentManager()  
 .popBackStack();  
 }  
 });  
 return view;  
 }  
  
 private void loadData() {  
 Bundle bundle = getArguments();  
 recordId = bundle.getInt("recordId");  
  
 Call<ServiceRecordViewModel> call = app.getStoService().getApi().getServiceRecord(recordId);  
 call.enqueue(new Callback<ServiceRecordViewModel>() {  
 @Override  
 public void onResponse(@NonNull Call<ServiceRecordViewModel> call, @NonNull Response<ServiceRecordViewModel> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 record = response.body();  
 record\_edit\_tv\_car\_brand\_and\_model.setText("Автомобиль: " + record.getCarBrandAndName());  
 String dateb = record.getDateBegin().split("T")[0] + " " + record.getDateBegin().split("T")[1].substring(0, 5);  
 record\_edit\_tv\_date\_begin.setText("Дата начала ТО: " + dateb);  
 dateb = record.getDateEnd().split("T")[0] + " " + record.getDateEnd().split("T")[1].substring(0, 5);  
 record\_edit\_tv\_date\_end.setText("Дата окончания ТО: " + dateb);  
 record\_edit\_editText\_description.setText(record.getDescription());  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<ServiceRecordViewModel> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void createOrUpdate() {  
 String description = String.valueOf(record\_edit\_editText\_description.getText());  
  
 ServiceRecordBindingModel bind\_record = new ServiceRecordBindingModel();  
 bind\_record.setCarId(record.getCarId());  
 bind\_record.setDateBegin(record.getDateBegin());  
 bind\_record.setDateEnd(record.getDateEnd());  
 bind\_record.setDescription(description);  
 bind\_record.setId(record.getId());  
  
 createOrUpdateRecord(bind\_record);  
 }  
  
 private void createOrUpdateRecord(ServiceRecordBindingModel bind\_record) {  
 Call<Void> call = app.getStoService().getApi().updateRecord(bind\_record);  
 call.enqueue(new Callback<Void>() {  
 @Override  
 public void onResponse(@NonNull Call<Void> call, @NonNull Response<Void> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Void> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
}

package com.example.myapplication.ui.to;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
  
import android.text.TextUtils;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.Spinner;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.models.bindingModels.CreateTOBindingModel;  
import com.example.myapplication.models.viewModels.CarViewModel;  
import com.example.myapplication.spinners.CarSpinnerAdapter;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class AddTOFragment extends Fragment {  
 private ArrayList<CarViewModel> cars;  
 private CarSpinnerAdapter adapter;  
 private Spinner spinner;  
 private App app;  
 private CarViewModel car;  
 private Button create\_button;  
 public AddTOFragment() {  
  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app =(App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_add\_to, container, false);  
 spinner = view.findViewById(R.id.add\_to\_spinner\_cars);  
 create\_button = view.findViewById(R.id.add\_to\_button\_create);  
 adapter = new CarSpinnerAdapter(getContext());  
 spinner.setAdapter(adapter);  
 restore();  
  
 spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {  
 car = (CarViewModel) adapterView.getItemAtPosition(i);  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> adapterView) {  
  
 }  
 });  
  
 create\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if (createWork()) {  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 getActivity().getSupportFragmentManager()  
 .popBackStack();  
  
 }  
 }  
 });  
 return view;  
 }  
  
 private void restore() {  
 //Подгружаем автомобили  
 Call<List<CarViewModel>> call = app.getStoService().getApi().getCars();  
 call.enqueue(new Callback<List<CarViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<CarViewModel>> call, @NonNull Response<List<CarViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 cars = (ArrayList<CarViewModel>) response.body();  
 if (cars != null) {  
 adapter.setData(cars);  
 adapter.notifyDataSetChanged();  
 }  
 else {  
 Toast.makeText(getContext(), "Привлеките клиентов", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<CarViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private boolean createWork() {  
 if (car != null) {  
 CreateTOBindingModel bind\_to = new CreateTOBindingModel();  
  
 bind\_to.setCarId(car.getId());  
 //TOdo: reg and auth  
 //bind\_to.setEmployeeId(App.getEmployee().getId());  
 bind\_to.setEmployeeId(1);  
 createWork(bind\_to);  
 return true;  
 }  
 else {  
 Toast.makeText(getContext(), "Введите данные", Toast.LENGTH\_SHORT).show();  
 return false;  
 }  
 }  
  
 private void createWork(CreateTOBindingModel model) {  
 Call<Void> call = app.getStoService().getApi().createTO(model);  
 call.enqueue(new Callback<Void>() {  
 @Override  
 public void onResponse(@NonNull Call<Void> call, @NonNull Response<Void> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Void> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
}

package com.example.myapplication.ui.to;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentTransaction;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.adapters.TOListAdapter;  
import com.example.myapplication.models.viewModels.TOViewModel;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class HomeFragment extends Fragment {  
 private TOListAdapter adapter;  
 private RecyclerView recyclerView;  
 private ArrayList<TOViewModel> tos;  
 private App app;  
 private FloatingActionButton fab;  
  
 public HomeFragment() {  
  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_home, container, false);  
 recyclerView = view.findViewById(R.id.recyclerview\_tos);  
  
 fab = view.findViewById(R.id.fab\_add\_to);  
  
 fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 AddTOFragment addTOFragment = new AddTOFragment();  
 replaceFragment(addTOFragment);  
 }  
 });  
  
 setRecyclerView();  
 // Inflate the layout for this fragment  
 return view;  
 }  
  
 private void setRecyclerView() {  
 recyclerView.setHasFixedSize(true);  
 recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));  
  
 adapter = new TOListAdapter(getContext());  
 recyclerView.setAdapter(adapter);  
 restore();  
  
 //при долгом нажатии на элемент открывается фрагмент его редактирования  
 adapter.setOnItemLongClickListener(new TOListAdapter.OnItemLongClickListener() {  
 @Override  
 public void onItemLongClicked(TOViewModel to, int position) {  
 Bundle bundle = new Bundle();  
 Integer id = to.getId();  
 bundle.putInt("toId", id);  
 bundle.putString("status", to.getStatus());  
 WatchTOFragment watchTOFragment = new WatchTOFragment();  
 watchTOFragment.setArguments(bundle);  
 replaceFragment(watchTOFragment);  
 }  
 });  
 }  
  
 private void restore() {  
 Call<List<TOViewModel>> call = app.getStoService().getApi().getTOs(App.getEmployee().getId());  
 call.enqueue(new Callback<List<TOViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<TOViewModel>> call, @NonNull Response<List<TOViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 tos = (ArrayList<TOViewModel>) response.body();  
 adapter.setData(tos);  
 adapter.notifyDataSetChanged();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<TOViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void replaceFragment(Fragment fragment) {  
 FragmentTransaction fragmentTransaction = getActivity()  
 .getSupportFragmentManager()  
 .beginTransaction();  
  
 fragmentTransaction.replace(R.id.frame\_layout, fragment);  
 fragmentTransaction.addToBackStack("");  
 fragmentTransaction.commit();  
 }  
}

package com.example.myapplication.ui.to;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.text.Html;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.adapters.WorkListAdapter;  
import com.example.myapplication.models.bindingModels.ChangeTOStatusBindingModel;  
import com.example.myapplication.models.viewModels.TOViewModel;  
import com.example.myapplication.models.viewModels.WorkViewModel;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
  
public class WatchTOFragment extends Fragment {  
 private ArrayList<WorkViewModel> works;  
 private RecyclerView recyclerView;  
 private WorkListAdapter adapter;  
 private TextView watch\_to\_idTV;  
 private TextView watch\_to\_carnameTV;  
 private TextView watch\_to\_statusTV;  
 private TextView watch\_to\_datecreateTV;  
 private TextView watch\_to\_datebeginTV;  
 private TextView watch\_to\_dateendTV;  
 private TextView watch\_to\_sumTV;  
 private TOViewModel to;  
 private Integer toId;  
 private App app;  
 private Button watch\_to\_button\_action;  
 Boolean flag = false;  
 public WatchTOFragment() {  
 // Required empty public constructor  
 }  
  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_watch\_t\_o, container, false);  
 recyclerView = view.findViewById(R.id.watch\_to\_recyclerview\_cars);  
 watch\_to\_idTV = view.findViewById(R.id.watch\_to\_idTV);  
 watch\_to\_carnameTV = view.findViewById(R.id.watch\_to\_carnameTV);  
 watch\_to\_statusTV = view.findViewById(R.id.watch\_to\_statusTV);  
 watch\_to\_datecreateTV = view.findViewById(R.id.watch\_to\_datecreateTV);  
 watch\_to\_datebeginTV = view.findViewById(R.id.watch\_to\_datebeginTV);  
 watch\_to\_dateendTV = view.findViewById(R.id.watch\_to\_dateendTV);  
 watch\_to\_sumTV = view.findViewById(R.id.watch\_to\_sumTV);  
 watch\_to\_button\_action = view.findViewById(R.id.watch\_to\_button\_action);  
  
 setRecyclerView();  
  
 watch\_to\_button\_action.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if (to != null) {  
 doActionWithTO();  
 }  
 }  
 });  
 return view;  
 }  
  
 private void setRecyclerView() {  
 Bundle bundle = getArguments();  
 if (bundle != null) {  
 setButtonDesc(bundle.getString("status"));  
 toId = bundle.getInt("toId");  
 }  
  
 recyclerView.setHasFixedSize(true);  
 recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));  
  
 adapter = new WorkListAdapter(getContext());  
 recyclerView.setAdapter(adapter);  
 restore();  
 }  
  
 private void restore() {  
 Call<List<WorkViewModel>> call = app.getStoService().getApi().getWorksByTO(toId);  
 call.enqueue(new Callback<List<WorkViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<WorkViewModel>> call, @NonNull Response<List<WorkViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 works = (ArrayList<WorkViewModel>) response.body();  
 adapter.setData(works);  
 adapter.notifyDataSetChanged();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<WorkViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
  
 Call<TOViewModel> call1 = app.getStoService().getApi().getTO(toId);  
 call1.enqueue(new Callback<TOViewModel>() {  
 @Override  
 public void onResponse(@NonNull Call<TOViewModel> call, @NonNull Response<TOViewModel> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 to = response.body();  
 watch\_to\_idTV.append(String.valueOf(to.getId()));  
 watch\_to\_carnameTV.append(to.getCarName());  
 watch\_to\_statusTV.setText(Html.fromHtml(getStatus(to.getStatus())), TextView.BufferType.SPANNABLE);  
 watch\_to\_datecreateTV.append(getDateNormalFormat(to.getDateCreate()));  
 watch\_to\_datebeginTV.append(getDateNormalFormat(to.getDateImplement()));  
 watch\_to\_dateendTV.append(getDateNormalFormat(to.getDateOver()));  
 watch\_to\_sumTV.append(String.valueOf(to.getSum()));  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<TOViewModel> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private String getStatus(String s) {  
 switch (s) {  
 case "Принят":  
 return "<font color='black'>Статус: </font><font color='red'>" + s + "</font>";  
 case "Выполняется":  
 return "<font color='black'>Статус: </font><font color='yellow'>" + s + "</font>";  
 case "Готов":  
 return "<font color='black'>Статус: </font><font color='blue'>" + s + "</font>";  
 default:  
 return "<font color='black'>Статус: </font><font color='green'>" + s + "</font>";  
 }  
 }  
  
 private String getDateNormalFormat(String s) {  
 if (!s.equals("-"))  
 return s.split("T")[0] + " " + s.split("T")[1].substring(0, 5);  
 return "-";  
 }  
  
 private boolean doActionWithTO() {  
 ChangeTOStatusBindingModel to\_bind = new ChangeTOStatusBindingModel();  
 to\_bind.setTOid(to.getId());  
  
 switch (to.getStatus()) {  
 case "Принят":  
 Call<Boolean> call1 = app.getStoService().getApi().takeTOInWork(to\_bind);  
 call1.enqueue(new Callback<Boolean>() {  
 @Override  
 public void onResponse(@NonNull Call<Boolean> call, @NonNull Response<Boolean> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 flag = Boolean.TRUE.equals(response.body());  
 Toast.makeText(getContext(), "ТО отправлено в работу", Toast.LENGTH\_SHORT).show();  
 if (flag) {  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 getActivity().getSupportFragmentManager()  
 .popBackStack();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Boolean> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 break;  
 case "Выполняется":  
 Call<Boolean> call2 = app.getStoService().getApi().finishTO(to\_bind);  
 call2.enqueue(new Callback<Boolean>() {  
 @Override  
 public void onResponse(@NonNull Call<Boolean> call, @NonNull Response<Boolean> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 flag = Boolean.TRUE.equals(response.body());  
 if (flag) {  
 Toast.makeText(getContext(), "ТО готово!", Toast.LENGTH\_SHORT).show();  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 getActivity().getSupportFragmentManager()  
 .popBackStack();  
 }  
 else  
 Toast.makeText(getContext(), "Не все работы завершены!", Toast.LENGTH\_SHORT).show();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Boolean> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 break;  
 case "Готов":  
 Call<Boolean> call3 = app.getStoService().getApi().issueTO(to\_bind);  
 call3.enqueue(new Callback<Boolean>() {  
 @Override  
 public void onResponse(@NonNull Call<Boolean> call, @NonNull Response<Boolean> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 flag = Boolean.TRUE.equals(response.body());  
 Toast.makeText(getContext(), "ТО выдано!", Toast.LENGTH\_SHORT).show();  
 if (flag) {  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 getActivity().getSupportFragmentManager()  
 .popBackStack();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Boolean> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 break;  
 }  
 return flag;  
 }  
  
 private void setButtonDesc(String s) {  
 switch (s) {  
 case "Выполняется":  
 watch\_to\_button\_action.setText("Завершить ТО");  
 break;  
 case "Готов":  
 watch\_to\_button\_action.setText("Выдать автомобиль");  
 break;  
 case "Выдан":  
 watch\_to\_button\_action.setEnabled(false);  
 watch\_to\_button\_action.setText("Выдано!");  
 break;  
 }  
 }  
}

package com.example.myapplication.ui.works;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
  
import android.text.TextUtils;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Spinner;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.models.bindingModels.CreateWorkBindingModel;  
import com.example.myapplication.models.viewModels.StoreKeeperViewModel;  
import com.example.myapplication.models.viewModels.TOViewModel;  
import com.example.myapplication.models.viewModels.WorkTypeViewModel;  
import com.example.myapplication.spinners.StoreKeeperSpinnerAdapter;  
import com.example.myapplication.spinners.TOSpinnerAdapter;  
import com.example.myapplication.spinners.WorkTypeSpinnerAdapter;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class CreateWorkFragment extends Fragment {  
 private ArrayList<TOViewModel> tos;  
 private ArrayList<WorkTypeViewModel> workTypes;  
 private ArrayList<StoreKeeperViewModel> storeKeepers;  
 private TOSpinnerAdapter adapterTO;  
 private WorkTypeSpinnerAdapter adapterWorkTypes;  
 private StoreKeeperSpinnerAdapter adapterStoreKeeper;  
 private Spinner spinnerTO;  
 private Spinner spinnerWorkType;  
 private Spinner spinnerStoreKeeper;  
 private App app;  
 private EditText create\_work\_countET;  
 private TextView create\_work\_priceET;  
 private TextView create\_work\_netpriceET;  
 private Button create\_work\_button\_work\_create;  
 private TOViewModel to;  
 private WorkTypeViewModel workType;  
 private StoreKeeperViewModel storeKeeper;  
 private Button create\_work\_button\_calc;  
  
 public CreateWorkFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_create\_work, container, false);  
 spinnerTO = view.findViewById(R.id.add\_to\_spinner\_cars);  
 spinnerWorkType = view.findViewById(R.id.create\_work\_spinner\_work\_types);  
 spinnerStoreKeeper = view.findViewById(R.id.create\_work\_spinner\_storekeeper);  
 create\_work\_countET = view.findViewById(R.id.watch\_work\_storekeeperfioTV);  
 create\_work\_priceET = view.findViewById(R.id.create\_work\_priceET);  
 create\_work\_netpriceET = view.findViewById(R.id.create\_work\_netpriceET);  
 create\_work\_button\_work\_create = view.findViewById(R.id.create\_work\_button\_work\_create);  
 create\_work\_button\_calc = view.findViewById(R.id.create\_work\_button\_calc);  
 initUI();  
  
 create\_work\_button\_work\_create.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if (createWork()) {  
 try {  
 Thread.sleep(400);  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 getActivity().getSupportFragmentManager()  
 .popBackStack();  
 }  
 }  
 });  
  
 create\_work\_button\_calc.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 calcNetPrice();  
 }  
 });  
  
 spinnerTO.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {  
 to = (TOViewModel) adapterView.getItemAtPosition(i);  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> adapterView) {  
  
 }  
 });  
  
 spinnerWorkType.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {  
 workType = (WorkTypeViewModel) adapterView.getItemAtPosition(i);  
 calcNetPrice();  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> adapterView) {  
  
 }  
 });  
  
 spinnerStoreKeeper.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {  
 storeKeeper = (StoreKeeperViewModel) adapterView.getItemAtPosition(i);  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> adapterView) {  
  
 }  
 });  
 return view;  
 }  
  
 private void initUI() {  
 adapterTO = new TOSpinnerAdapter(getContext());  
 adapterWorkTypes = new WorkTypeSpinnerAdapter(getContext());  
 adapterStoreKeeper = new StoreKeeperSpinnerAdapter(getContext());  
 spinnerTO.setAdapter(adapterTO);  
 spinnerWorkType.setAdapter(adapterWorkTypes);  
 spinnerStoreKeeper.setAdapter(adapterStoreKeeper);  
  
 restore();  
 }  
  
 private void restore() {  
 //Подгружаем ТОшки  
 Call<List<TOViewModel>> call = app.getStoService().getApi().getNotStartedTOs(App.getEmployee().getId());  
 call.enqueue(new Callback<List<TOViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<TOViewModel>> call, @NonNull Response<List<TOViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 tos = (ArrayList<TOViewModel>) response.body();  
 if (tos != null) {  
 adapterTO.setData(tos);  
 adapterTO.notifyDataSetChanged();  
 }  
 else {  
 Toast.makeText(getContext(), "Создайте ТО", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<TOViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
  
 //Подгружаем типы услуг  
 Call<List<WorkTypeViewModel>> call2 = app.getStoService().getApi().getWorkTypes();  
 call2.enqueue(new Callback<List<WorkTypeViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<WorkTypeViewModel>> call, @NonNull Response<List<WorkTypeViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
 workTypes = (ArrayList<WorkTypeViewModel>) response.body();  
 if (workTypes != null) {  
 adapterWorkTypes.setData(workTypes);  
 adapterWorkTypes.notifyDataSetChanged();  
 }  
 else {  
 Toast.makeText(getContext(), "Создайте типы услуг", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<WorkTypeViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
  
 //Подгружаем типы кладовщиков  
 Call<List<StoreKeeperViewModel>> call3 = app.getStoService().getApi().getStoreKeepers();  
 call3.enqueue(new Callback<List<StoreKeeperViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<StoreKeeperViewModel>> call, @NonNull Response<List<StoreKeeperViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
 storeKeepers = (ArrayList<StoreKeeperViewModel>) response.body();  
 if (storeKeepers != null) {  
 adapterStoreKeeper.setData(storeKeepers);  
 adapterStoreKeeper.notifyDataSetChanged();  
 }  
 else {  
 Toast.makeText(getContext(), "Возьмите на работу кладовщиков", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<StoreKeeperViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private boolean createWork() {  
 if (!TextUtils.isEmpty(create\_work\_countET.getText()) &&  
 !TextUtils.isEmpty(create\_work\_priceET.getText()) &&  
 !TextUtils.isEmpty(create\_work\_netpriceET.getText()) &&  
 to != null && workType != null && storeKeeper != null) {  
  
 Integer count = Integer.valueOf(String.valueOf(create\_work\_countET.getText()));  
 Float price = Float.valueOf(String.valueOf(create\_work\_priceET.getText()));  
 Float netprice = Float.valueOf(String.valueOf(create\_work\_netpriceET.getText()));  
  
 CreateWorkBindingModel bind\_work = new CreateWorkBindingModel();  
  
 bind\_work.setCount(count);  
 bind\_work.setPrice(price);  
 bind\_work.setNetPrice(netprice);  
 bind\_work.setTOId(to.getId());  
 bind\_work.setWorkTypeId(workType.getId());  
 bind\_work.setStoreKeeperId(storeKeeper.getId());  
  
 createWork(bind\_work);  
 return true;  
 }  
 else {  
 Toast.makeText(getContext(), "Введите данные", Toast.LENGTH\_SHORT).show();  
 return false;  
 }  
 }  
  
 private void createWork(CreateWorkBindingModel model) {  
 Call<Void> call = app.getStoService().getApi().createWork(model);  
 call.enqueue(new Callback<Void>() {  
 @Override  
 public void onResponse(@NonNull Call<Void> call, @NonNull Response<Void> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<Void> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void calcNetPrice() {  
 int k = tryParseInt(String.valueOf(create\_work\_countET.getText()));  
 if (k != 0 && workType != null) {  
  
 Float price = workType.getPrice() \* k;  
 Float netprice = workType.getNetPrice() \* k;  
 create\_work\_priceET.setText(String.valueOf(price));  
 create\_work\_netpriceET.setText(String.valueOf(netprice));  
 }  
 }  
  
 private int tryParseInt(String value) {  
 try {  
 return Integer.parseInt(value);  
 } catch (NumberFormatException ex) {  
 return 0;  
 }  
 }  
}

package com.example.myapplication.ui.works;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.text.Html;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.adapters.SparePartListAdapter;  
import com.example.myapplication.models.viewModels.SparePartViewModel;  
import com.example.myapplication.models.viewModels.WorkTypeViewModel;  
import com.example.myapplication.models.viewModels.WorkViewModel;  
import com.example.myapplication.models.webViewModels.SparePartWebViewModel;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class WatchWorkFragment extends Fragment {  
 private ArrayList<SparePartWebViewModel> parts;  
 private RecyclerView recyclerView;  
 private SparePartListAdapter adapter;  
 private TextView watch\_work\_storekeeperfioTV;  
 private TextView watch\_work\_worknameTV;  
 private TextView watch\_work\_priceTV;  
 private TextView watch\_work\_netpriceTV;  
 private TextView watch\_work\_statusTV;  
 private TextView watch\_work\_workbeginTV;  
 private TextView watch\_work\_countTV;  
 private WorkViewModel work;  
 private Integer workId;  
 private App app;  
  
 public WatchWorkFragment() {  
 // Required empty public constructor  
 }  
  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_watch\_work, container, false);  
 recyclerView = view.findViewById(R.id.recyclerview\_spareparts);  
 watch\_work\_storekeeperfioTV = view.findViewById(R.id.watch\_work\_storekeeperfioTV);  
 watch\_work\_worknameTV = view.findViewById(R.id.watch\_work\_worknameTV);  
 watch\_work\_priceTV = view.findViewById(R.id.watch\_work\_priceTV);  
 watch\_work\_netpriceTV = view.findViewById(R.id.watch\_work\_netpriceTV);  
 watch\_work\_statusTV = view.findViewById(R.id.watch\_work\_statusTV);  
 watch\_work\_workbeginTV = view.findViewById(R.id.watch\_work\_workbeginTV);  
 watch\_work\_countTV = view.findViewById(R.id.watch\_work\_countTV);  
  
 setRecyclerView();  
  
 return view;  
 }  
  
 private void setRecyclerView() {  
 Bundle bundle = getArguments();  
 if (bundle != null)  
 workId = bundle.getInt("workId");  
 recyclerView.setHasFixedSize(true);  
 recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));  
  
 adapter = new SparePartListAdapter(getContext());  
 recyclerView.setAdapter(adapter);  
 restore();  
 }  
  
 private void restore() {  
 if (workId != null) {  
 Call<WorkViewModel> call1 = app.getStoService().getApi().getWork(workId);  
 call1.enqueue(new Callback<WorkViewModel>() {  
 @Override  
 public void onResponse(@NonNull Call<WorkViewModel> call, @NonNull Response<WorkViewModel> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 work = response.body();  
 watch\_work\_storekeeperfioTV.append(String.valueOf(work.getStoreKeeperFIO()));  
 watch\_work\_worknameTV.append(work.getWorkName());  
 watch\_work\_priceTV.append(String.valueOf(work.getPrice()));  
 watch\_work\_netpriceTV.append(String.valueOf(work.getNetPrice()));  
 watch\_work\_statusTV.setText(Html.fromHtml(getStatus(work.getWorkStatus())), TextView.BufferType.SPANNABLE);  
 watch\_work\_workbeginTV.append(getDateNormalFormat(work.getWorkBegin()));  
 watch\_work\_countTV.append(String.valueOf(work.getCount()));  
 loadParts();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<WorkViewModel> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
 }  
  
 private void loadParts() {  
 if (work != null) {  
 Call<List<SparePartWebViewModel>> call = app.getStoService().getApi().getSparePartsByWorkType(work.getWorkTypeId());  
 call.enqueue(new Callback<List<SparePartWebViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<SparePartWebViewModel>> call, @NonNull Response<List<SparePartWebViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 parts = (ArrayList<SparePartWebViewModel>) response.body();  
 adapter.setData(parts);  
 adapter.notifyDataSetChanged();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<SparePartWebViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
 }  
  
 private String getStatus(String s) {  
 switch (s) {  
 case "Принят":  
 return "<font color='black'>Статус: </font><font color='red'>" + s + "</font>";  
 case "Выполняется":  
 return "<font color='black'>Статус: </font><font color='yellow'>" + s + "</font>";  
 case "Готов":  
 return "<font color='black'>Статус: </font><font color='blue'>" + s + "</font>";  
 default:  
 return "<font color='black'>Статус: </font><font color='green'>" + s + "</font>";  
 }  
 }  
  
 private String getDateNormalFormat(String s) {  
 if (!s.equals("-"))  
 return s.split("T")[0] + " " + s.split("T")[1].substring(0, 5);  
 return "Не в работе";  
 }  
}

package com.example.myapplication.ui.works;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentTransaction;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.R;  
import com.example.myapplication.adapters.WorkListAdapter;  
import com.example.myapplication.models.viewModels.WorkViewModel;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
  
public class WorksFragment extends Fragment {  
 private WorkListAdapter adapter;  
 private RecyclerView recyclerView;  
 private ArrayList<WorkViewModel> works;  
 private App app;  
 private FloatingActionButton fab;  
  
 public WorksFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_works, container, false);  
 recyclerView = view.findViewById(R.id.recyclerview\_works);  
  
 fab = view.findViewById(R.id.fab\_add\_work);  
  
 fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 CreateWorkFragment createWorkFragment = new CreateWorkFragment();  
 replaceFragment(createWorkFragment);  
 }  
 });  
  
 setRecyclerView();  
 // Inflate the layout for this fragment  
 return view;  
 }  
  
  
 private void setRecyclerView() {  
 recyclerView.setHasFixedSize(true);  
 recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));  
  
 adapter = new WorkListAdapter(getContext());  
 recyclerView.setAdapter(adapter);  
 restore();  
  
 //при долгом нажатии на элемент открывается фрагмент его редактирования  
 adapter.setOnItemLongClickListener(new WorkListAdapter.OnItemLongClickListener() {  
 @Override  
 public void onItemLongClicked(WorkViewModel work, int position) {  
 Bundle bundle = new Bundle();  
 Integer id = work.getId();  
 bundle.putInt("workId", id);  
 WatchWorkFragment watchWorkFragment = new WatchWorkFragment();  
 watchWorkFragment.setArguments(bundle);  
 replaceFragment(watchWorkFragment);  
 }  
 });  
 }  
  
 private void restore() {  
 Call<List<WorkViewModel>> call = app.getStoService().getApi().getWorksByEmployee(App.getEmployee().getId());  
 call.enqueue(new Callback<List<WorkViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<WorkViewModel>> call, @NonNull Response<List<WorkViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 works = (ArrayList<WorkViewModel>) response.body();  
 adapter.setData(works);  
 adapter.notifyDataSetChanged();  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<WorkViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void replaceFragment(Fragment fragment) {  
 FragmentTransaction fragmentTransaction = getActivity()  
 .getSupportFragmentManager()  
 .beginTransaction();  
  
 fragmentTransaction.replace(R.id.frame\_layout, fragment);  
 fragmentTransaction.addToBackStack("");  
 fragmentTransaction.commit();  
 }  
}

package com.example.myapplication.ui;  
  
import android.content.Intent;  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
  
import android.os.Environment;  
import android.text.TextUtils;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Spinner;  
import android.widget.Toast;  
  
import com.example.myapplication.App;  
import com.example.myapplication.LogInRegisterActivity;  
import com.example.myapplication.R;  
import com.example.myapplication.models.bindingModels.EmployeeBindingModel;  
import com.example.myapplication.models.viewModels.CarViewModel;  
import com.example.myapplication.models.viewModels.ServiceRecordViewModel;  
import com.example.myapplication.models.viewModels.TOViewModel;  
import com.example.myapplication.models.viewModels.WorkTypeViewModel;  
import com.example.myapplication.spinners.CarSpinnerAdapter;  
import com.itextpdf.text.Document;  
import com.itextpdf.text.DocumentException;  
import com.itextpdf.text.Element;  
import com.itextpdf.text.Font;  
import com.itextpdf.text.Paragraph;  
import com.itextpdf.text.Phrase;  
import com.itextpdf.text.pdf.BaseFont;  
import com.itextpdf.text.pdf.PdfPTable;  
import com.itextpdf.text.pdf.PdfWriter;  
  
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Map;  
  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
  
public class ProfileFragment extends Fragment {  
 private EditText profile\_fioTV;  
 private EditText profile\_loginTV;  
 private EditText profile\_passwordTV;  
 private Button profile\_button\_update\_data;  
 private Button profile\_button\_excel\_report;  
 private Button profile\_button\_word\_report;  
 private Button profile\_button\_pdf\_report;  
 private Button profile\_button\_exit;  
 private App app;  
 private ArrayList<CarViewModel> cars;  
 private CarSpinnerAdapter adapter;  
 private Spinner spinner;  
 private CarViewModel car;  
 public ProfileFragment() {  
 // Required empty public constructor  
 }  
  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 app = (App) getActivity().getApplication();  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.fragment\_profile, container, false);  
 spinner = view.findViewById(R.id.profile\_spinner\_cars);  
 profile\_fioTV = view.findViewById(R.id.profile\_fioTV);  
 profile\_loginTV = view.findViewById(R.id.profile\_loginTV);  
 profile\_passwordTV = view.findViewById(R.id.profile\_passwordTV);  
 profile\_button\_update\_data = view.findViewById(R.id.profile\_button\_update\_data);  
 profile\_button\_excel\_report = view.findViewById(R.id.profile\_button\_excel\_report);  
 profile\_button\_word\_report = view.findViewById(R.id.profile\_button\_word\_report);  
 profile\_button\_pdf\_report = view.findViewById(R.id.profile\_button\_pdf\_report);  
 profile\_button\_exit = view.findViewById(R.id.profile\_button\_exit);  
  
 adapter = new CarSpinnerAdapter(getContext());  
 spinner.setAdapter(adapter);  
  
 loadData();  
  
 spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {  
 car = (CarViewModel) adapterView.getItemAtPosition(i);  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> adapterView) {  
  
 }  
 });  
  
 profile\_button\_update\_data.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 updateProfile();  
 }  
 });  
  
 profile\_button\_exit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 App.setEmployee(null);  
 Intent intent = new Intent(getActivity(), LogInRegisterActivity.class);  
 startActivity(intent);  
 }  
 });  
  
 profile\_button\_excel\_report.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 createReportWorks();  
 }  
 });  
  
 profile\_button\_word\_report.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 createReportAuto();  
 }  
 });  
  
 profile\_button\_pdf\_report.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 createReportTOs();  
 }  
 });  
 return view;  
 }  
  
 private void loadData() {  
 profile\_fioTV.setText(App.getEmployee().getFIO());  
 profile\_loginTV.setText(App.getEmployee().getLogin());  
 profile\_passwordTV.setText(App.getEmployee().getPassword());  
  
 //Подгружаем автомобили  
 Call<List<CarViewModel>> call = app.getStoService().getApi().getCars();  
 call.enqueue(new Callback<List<CarViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<CarViewModel>> call, @NonNull Response<List<CarViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 cars = (ArrayList<CarViewModel>) response.body();  
 if (cars != null) {  
 adapter.setData(cars);  
 adapter.notifyDataSetChanged();  
 }  
 else {  
 Toast.makeText(getContext(), "Привлеките клиентов", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<CarViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void updateProfile() {  
 if (!TextUtils.isEmpty(profile\_fioTV.getText())  
 && !TextUtils.isEmpty(profile\_loginTV.getText()) &&  
 !TextUtils.isEmpty(profile\_passwordTV.getText())) {  
 EmployeeBindingModel model = new EmployeeBindingModel();  
 model.setId(App.getEmployee().getId());  
 model.setFIO(String.valueOf(profile\_fioTV.getText()));  
 model.setLogin(String.valueOf(profile\_loginTV.getText()));  
 model.setPassword(String.valueOf(profile\_passwordTV.getText()));  
 Call<Void> call = app.getStoService().getApi().updateProfile(model);  
  
 call.enqueue(new Callback<Void>() {  
 @Override  
 public void onResponse(Call<Void> call, Response<Void> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 App.getEmployee().setFIO(String.valueOf(profile\_fioTV.getText()));  
 App.getEmployee().setLogin(String.valueOf(profile\_loginTV.getText()));  
 App.getEmployee().setPassword(String.valueOf(profile\_passwordTV.getText()));  
 Toast.makeText(getContext(), "Данные изменены", Toast.LENGTH\_SHORT).show();  
 }  
  
 @Override  
 public void onFailure(Call<Void> call, Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
 }  
  
 //Создание отчёта по ТО, сделанного работником  
 private void createReportTOs() {  
 Call<List<TOViewModel>> call = app.getStoService().getApi().getTOs(App.getEmployee().getId());  
 call.enqueue(new Callback<List<TOViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<TOViewModel>> call, @NonNull Response<List<TOViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 ArrayList<TOViewModel> tos = (ArrayList<TOViewModel>) response.body();  
 try {  
 if (tos != null && tos.size() > 0)  
 createPdfReportTOS(tos);  
 else  
 Toast.makeText(getContext(), "Проведите хоть одно ТО", Toast.LENGTH\_SHORT).show();  
 } catch (IOException | DocumentException e) {  
 e.printStackTrace();  
 Toast.makeText(getContext(), "Ошибка создания отчёта", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<TOViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void createPdfReportTOS( ArrayList<TOViewModel> tos) throws IOException, DocumentException {  
 String filename = App.getEmployee().getFIO() + " ТО.pdf";  
 Document document = new Document();  
 File root = new File(getContext().getExternalFilesDir(Environment.DIRECTORY\_DOWNLOADS), "Reports");  
  
 if (!root.exists()) {  
 root.mkdirs();  
 }  
  
 BaseFont bf = BaseFont.createFont("/assets/fonts/arial.ttf", BaseFont.IDENTITY\_H, BaseFont.EMBEDDED);  
  
 Font font = new Font(bf, 12, Font.NORMAL);  
  
 File gpxfile = new File(root, filename); // generate pdf file in that directory  
  
 PdfWriter.getInstance(document,new FileOutputStream(gpxfile));  
  
 document.open();  
  
 Paragraph p = new Paragraph("ТО работника \"" + App.getEmployee().getFIO() + "\"", font);  
 document.add(p);  
  
 Paragraph p2 = new Paragraph();  
 p2.add(" ");  
 document.add(p2);  
  
 Paragraph p3 = new Paragraph();  
 p3.add(" ");  
 document.add(p3);  
  
  
 PdfPTable tablePdf = new PdfPTable(7);  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Номер ТО", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Автомобиль", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Статус", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Дата создания", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Дата начала", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Дата окончания", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Сумма", font));  
  
 for(TOViewModel to : tos){  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, String.valueOf(to.getId()), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, to.getCarName(), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, to.getStatus(), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, getDateNormalFormat(to.getDateCreate()), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, getDateNormalFormat(to.getDateImplement()), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, getDateNormalFormat(to.getDateOver()), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, String.valueOf(to.getSum()), font));  
 }  
  
 document.add(tablePdf);  
 document.addCreationDate();  
 document.close();  
 Toast.makeText(getContext(), "Файл успешно создан", Toast.LENGTH\_SHORT).show();  
 }  
  
 //Создание отчёта по всем услугам  
 private void createReportWorks() {  
 Call<List<WorkTypeViewModel>> call = app.getStoService().getApi().getWorkTypes();  
 call.enqueue(new Callback<List<WorkTypeViewModel>>() {  
 @Override  
 public void onResponse(@NonNull Call<List<WorkTypeViewModel>> call, @NonNull Response<List<WorkTypeViewModel>> response) {  
 if (!response.isSuccessful()) {  
 Toast.makeText(getContext(), response.code(), Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 ArrayList<WorkTypeViewModel> works = (ArrayList<WorkTypeViewModel>) response.body();  
 try {  
 if (works != null && works.size() > 0)  
 createPdfReportWorks(works);  
 else  
 Toast.makeText(getContext(), "Создайте хоть одну услугу", Toast.LENGTH\_SHORT).show();  
 } catch (IOException | DocumentException e) {  
 e.printStackTrace();  
 Toast.makeText(getContext(), "Ошибка создания отчёта", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @Override  
 public void onFailure(@NonNull Call<List<WorkTypeViewModel>> call, @NonNull Throwable t) {  
 Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH\_SHORT).show();  
 Log.d("BBB", t.getMessage());  
 }  
 });  
 }  
  
 private void createPdfReportWorks(ArrayList<WorkTypeViewModel> works) throws IOException, DocumentException {  
 //Получение запчасти по типу услуги  
 String filename = "услуги.pdf";  
 Document document = new Document();  
 File root = new File(getContext().getExternalFilesDir(Environment.DIRECTORY\_DOWNLOADS), "Reports");  
  
 if (!root.exists()) {  
 root.mkdirs();  
 }  
  
 BaseFont bf = BaseFont.createFont("/assets/fonts/arial.ttf", BaseFont.IDENTITY\_H, BaseFont.EMBEDDED);  
  
 Font font = new Font(bf, 12, Font.NORMAL);  
  
 File gpxfile = new File(root, filename); // generate pdf file in that directory  
  
 PdfWriter.getInstance(document,new FileOutputStream(gpxfile));  
  
 document.open();  
  
 Paragraph p = new Paragraph("Наши услуги ", font);  
 document.add(p);  
  
 Paragraph p2 = new Paragraph();  
 p2.add(" ");  
 document.add(p2);  
  
 Paragraph p3 = new Paragraph();  
 p3.add(" ");  
 document.add(p3);  
  
  
 PdfPTable tablePdf = new PdfPTable(6);  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Номер услуги", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Название", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Время выполнения", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Цена услуги", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Цена с уч. запчастей", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Запчасти (кол-во)", font));  
 for(WorkTypeViewModel work : works){  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, String.valueOf(work.getId()), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, work.getWorkName(), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, work.getExecutionTime(), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, String.valueOf(work.getPrice()), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, String.valueOf(work.getNetPrice()), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, String.valueOf(work.getWorkSpareParts().size()), font));  
 }  
  
 document.add(tablePdf);  
 document.addCreationDate();  
 document.close();  
 Toast.makeText(getContext(), "Файл успешно создан", Toast.LENGTH\_SHORT).show();  
 }  
  
 //Создание отчёта по автомобилю  
 private void createReportAuto() {  
 if (car != null) {  
 try {  
 createPdfReportAuto();  
 } catch (IOException | DocumentException e) {  
 e.printStackTrace();  
 Toast.makeText(getContext(), "Ошибка создания отчёта", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 else {  
 Toast.makeText(getContext(), "Выберите автомобиль", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 private void createPdfReportAuto() throws IOException, DocumentException {  
 String filename = car.getBrand() + " " + car.getModel() + ".pdf";  
 Document document = new Document();  
 File root = new File(getContext().getExternalFilesDir(Environment.DIRECTORY\_DOWNLOADS), "Reports");  
  
 if (!root.exists()) {  
 root.mkdirs();  
 }  
  
 BaseFont bf = BaseFont.createFont("/assets/fonts/arial.ttf", BaseFont.IDENTITY\_H, BaseFont.EMBEDDED);  
  
 Font font = new Font(bf, 12, Font.NORMAL);  
  
 File gpxfile = new File(root, filename); // generate pdf file in that directory  
  
 PdfWriter.getInstance(document,new FileOutputStream(gpxfile));  
  
 document.open();  
  
 Paragraph p = new Paragraph("Автомобиль \"" + car.getBrand() + " " + car.getModel() + "\"", font);  
 document.add(p);  
  
 Paragraph p2 = new Paragraph();  
 p2.add(" ");  
 document.add(p2);  
  
 Paragraph p3 = new Paragraph();  
 p3.add(" ");  
 document.add(p3);  
  
  
 PdfPTable tablePdf = new PdfPTable(4);  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Марка авто", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Модель авто", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"VIN", font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT,"Номер телефона владельца", font));  
  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, String.valueOf(car.getBrand()), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, car.getModel(), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, car.getVIN(), font));  
 tablePdf.addCell(new Phrase(Element.ALIGN\_LEFT, car.getOwnerPhoneNumber(), font));  
  
 document.add(tablePdf);  
  
 Paragraph p4 = new Paragraph();  
 p4.add(" ");  
 document.add(p4);  
  
 Paragraph p5 = new Paragraph();  
 p5.add(" ");  
 document.add(p5);  
  
 Paragraph p6 = new Paragraph();  
 p4.add("Записи");  
 document.add(p6);  
  
 for(Map.Entry<Integer, String> record : car.getRecords().entrySet()){  
 Paragraph p7 = new Paragraph();  
 p7.add(record.getValue());  
 document.add(p7);  
 }  
  
 document.addCreationDate();  
 document.close();  
 Toast.makeText(getContext(), "Файл успешно создан", Toast.LENGTH\_SHORT).show();  
 }  
  
 private String getDateNormalFormat(String s) {  
 if (!s.equals("-"))  
 return s.split("T")[0] + " " + s.split("T")[1].substring(0, 5);  
 return "-";  
 }  
}

package com.example.myapplication;  
  
import android.app.Application;  
  
import com.example.myapplication.models.viewModels.EmployeeViewModel;  
import com.example.myapplication.service.STOService;  
  
public class App extends Application {  
  
 private STOService stoService;  
 public static EmployeeViewModel Employee;  
  
 public static EmployeeViewModel getEmployee() {  
 return Employee;  
 }  
  
 public static void setEmployee(EmployeeViewModel employee) {  
 Employee = employee;  
 }  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
  
 stoService = new STOService();  
 }  
  
 public STOService getStoService() {  
 return stoService;  
 }  
}

package com.example.myapplication;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentManager;  
import androidx.fragment.app.FragmentTransaction;  
  
import android.os.Bundle;  
  
import com.example.myapplication.ui.logregfragments.LogInFragment;  
  
public class LogInRegisterActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_log\_in\_register);  
  
 replaceFragment(new LogInFragment());  
 }  
  
 private void replaceFragment(Fragment fragment) {  
 FragmentManager fragmentManager = getSupportFragmentManager();  
 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();  
  
 fragmentTransaction.replace(R.id.frame\_layout\_entry, fragment);  
 fragmentTransaction.commit();  
 }  
}

package com.example.myapplication;  
  
import android.app.DownloadManager;  
import android.content.Context;  
import android.net.Uri;  
import android.os.Bundle;  
  
import com.example.myapplication.databinding.ActivityMainBinding;  
import com.example.myapplication.ui.cars.CarsFragment;  
import com.example.myapplication.ui.to.HomeFragment;  
import com.example.myapplication.ui.ProfileFragment;  
import com.example.myapplication.ui.works.WorksFragment;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentManager;  
import androidx.fragment.app.FragmentTransaction;  
  
public class MainActivity extends AppCompatActivity {  
  
 private ActivityMainBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 getWindow().setBackgroundDrawable(null);  
 binding = ActivityMainBinding.inflate(getLayoutInflater());  
 setContentView(binding.getRoot());  
 replaceFragment(new HomeFragment());  
  
 binding.navView.setOnItemSelectedListener(item -> {  
 switch (item.getItemId()) {  
 case R.id.navigation\_home:  
 replaceFragment(new HomeFragment());  
 break;  
 case R.id.navigation\_cars:  
 replaceFragment(new CarsFragment());  
 break;  
 case R.id.navigation\_works:  
 replaceFragment(new WorksFragment());  
 break;  
 case R.id.navigation\_profile:  
 replaceFragment(new ProfileFragment());  
 //TOdo: file download  
// DownloadManager manager = (DownloadManager) getSystemService(Context.DOWNLOAD\_SERVICE);  
// DownloadManager.Request request = new DownloadManager.Request(Uri.parse("http://192.168.0.127:7252/api/getfile.xlsx"));  
// request.setDescription("Test");  
// manager.enqueue(request);  
 break;  
 }  
 return true;  
 });  
 }  
  
 private void replaceFragment(Fragment fragment) {  
 FragmentManager fragmentManager = getSupportFragmentManager();  
 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();  
  
 fragmentTransaction.replace(R.id.frame\_layout, fragment);  
 fragmentTransaction.commit();  
 }  
}