Введение.

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

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

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

1.Техническое задание

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

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

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

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

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

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

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

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

* 1. Функциональные требования

1) Система регистрации и авторизации

2) Предоставление информации по техническому обслуживанию и ремонту автомобиля.

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

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

1.2 Нефункциональные требования

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

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

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

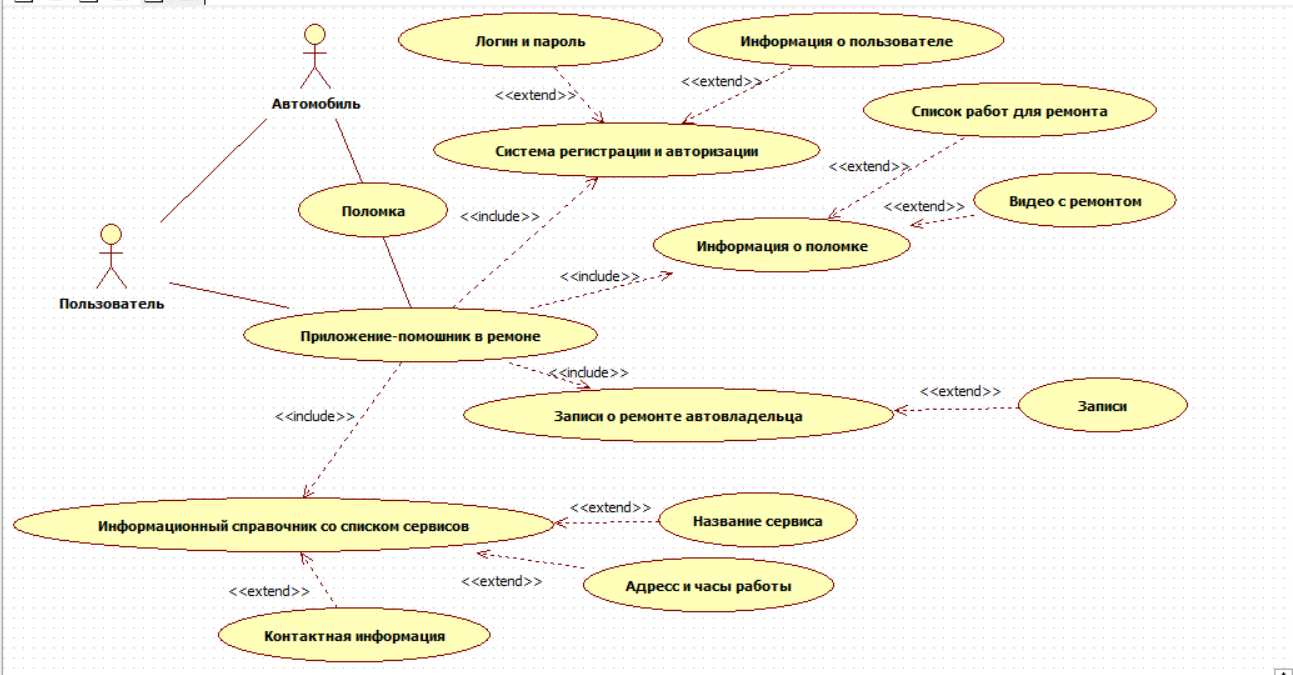
5)Интуитивно понятный и простой интерфейс.

2.Проектирование.

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

2.1 Диаграмма прецедентов.

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

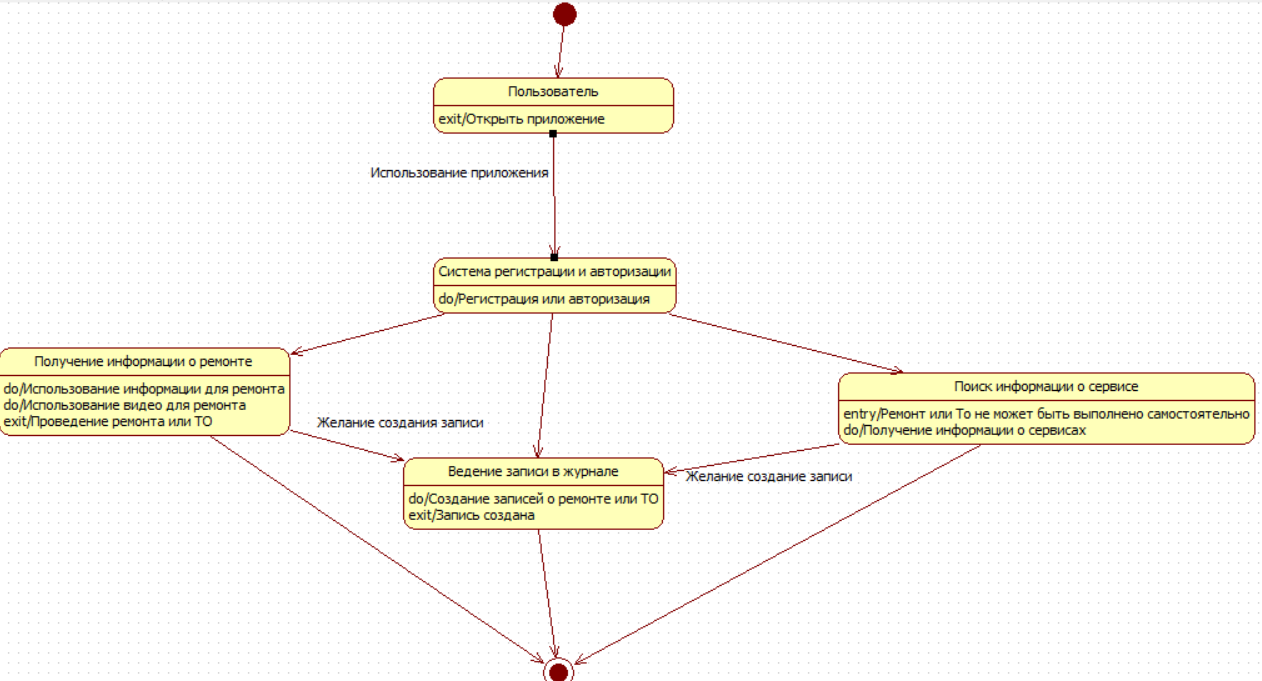
 Рис.1 Диаграмма прецедентов

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

Записи отмечены как расширение для системы записей о ремонте автомобиля. Контактная информация, адрес и название сервиса отмечены как расширения информационного справочника. Эта диаграмма позволяет понять, как работает функционал приложения. Пользователь, имеющий автомобиль, который сломался, может воспользоваться приложением, которое поможет ему с ремонтом. В приложении он может получить информацию о ремонте, сделать записи, а также найти подходящий сервис в случае, если произвести ремонт самостоятельно не представляется возможным.

2.2 Диаграмма состояний.

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

 Рис.2 Диаграмма состояний.

На данной диаграмме отображены состояния, которые требуют определённых действий, для перехода в следующее состояние. В случае разрабатываемой