МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Вятский государственный университет»
(ФГБОУ ВПО «ВятГУ»)
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Допущено к защите
Руководитель проекта
_____ (Клюкин В. Л.)
«___» _____2022г.

«Интерактивный помощник для игры Assetto Corsa Competizione»

Пояснительная записка курсового проекта по дисциплине
«Курс знаний бакалавра»
ТПЖА 090301.387 ПЗ

Разработал студент группы ИВТ-31 _____/Жеребцов К.А./

Руководитель _____/Клюкин В. Л./

Консультант _____/Долженкова М.Л./

Работа защищена с оценкой «_____» _____
(оценка) *(дата)*

Члены комиссии _____ /_____/
(подпись)
_____ /_____/
(подпись)
_____ /_____/
(подпись)

Киров 2022

Реферат

Жеребцов К. А. Разработка программного продукта «Интерактивный помощник для игры Assetto Corsa Competizione». ТПЖА. 090301.387 ПЗ: Курс. проект / ВятГУ, каф. ЭВМ; рук. Клюкин В. Л. - Киров, 2022. Графическая часть 3 л. – ф. А2, ПЗ 72 с, 21 рис., 3 табл., 5 источников, 4 приложений.

Разработка программного продукта «Интерактивный помощник для игры Assetto Corsa Competizione.

Объект исследования и разработки – автосимулятор Assetto Corsa Competizione.

Цель курсового проекта – разработка рабочего прототипа программного продукта.

Результат работы – готовый прототип программного продукта.

Содержание

Введение

С развитием компьютерных технологий и игровой индустрии в последние десятилетия появились симуляторы, которые поражают своей реалистичностью и глубиной геймплея. Одним из ярких представителей этого жанра является автосимулятор Assetto Corsa Competizione (ACC), который позволяет игрокам погрузиться в мир автоспорта, почувствовать азарт соревнований и отточить свои навыки вождения.

В данном курсовом проекте мы рассмотрим основные аспекты создания и функционал ACC Helper, подчеркивая важность данного приложения в контексте современной игровой индустрии. Мы проанализируем технические аспекты его разработки, а также рассмотрим потенциал, который он открывает для геймеров и сообщества поклонников автоспорта.

1. Анализ предметной области

В данном разделе рассматриваются обоснование актуальности выбранной темы и аналоги, представленные существующим программным обеспечением.

1.1 Обоснование актуальности темы

Развитие технологий в игровой индустрии и появление реалистичных автосимуляторов, таких как Assetto Corsa Competizione (ACC), привели к растущему интересу к миру виртуальных гоночных соревнований. С каждым годом игроки становятся все более преданными и стремятся улучшить свои навыки вождения, совершенствовать стратегии и участвовать в соревнованиях.

В этом контексте приложения, способные помогать игрокам оптимизировать свои стратегии, а также предоставлять информацию о текущих турнирах и мероприятиях, становятся крайне актуальными. ACC Helper предлагает не только техническую помощь в выборе оптимальной стратегии на гонку, но и обеспечивает возможность отслеживать актуальные соревнования в игре.

Актуальность данного приложения обусловлена несколькими факторами:

- Оптимизация игрового опыта: ACC Helper позволяет игрокам улучшить свои навыки, предоставляя аналитику, основанную на данных, которые трудно получить внутри самой игры. Это помогает игрокам быстрее учиться и совершенствовать свои навыки, делая игровой опыт более увлекательным и интересным.
- Соревновательный аспект: Игроки всегда стремятся участвовать в турнирах и состязаниях, чтобы продемонстрировать свои навыки и получить признание. ACC Helper обеспечивает игроков актуальной информацией о турнирах, что позволяет им активно участвовать в

соревнованиях и быть в курсе всех актуальных событий в игровом мире.

Таким образом, ACC Helper не только облегчает игровой процесс, но и поднимает социальный и соревновательный аспект игрового мира, делая его более интересным и захватывающим для многих игроков.

1.2 Обзор аналогов

существуют различные аналоги приложений, предназначенных для помощи игрокам в автосимуляторах и в поддержке их стратегий. Некоторые из них предоставляют аналогичные функции, как и ACC Helper, включая анализ данных, стратегические советы и отслеживание соревнований. Вот несколько примеров аналогичных приложений:

1. **iRacing:** iRacing - это онлайн-платформа для гоночных симуляторов, которая предоставляет высококачественные гоночные симуляторы и соревнования. Она предлагает аналитические инструменты, позволяющие игрокам анализировать свои технические данные и стратегии гонок.
2. **RaceRoom:** RaceRoom - это гоночный симулятор с множеством настроек и возможностей. Он также предоставляет отслеживание результатов, аналитику и обсуждение стратегий через свое сообщество.
3. **Rfactor 2:** Rfactor 2 - это реалистичный гоночный симулятор с активным сообществом игроков. Существует несколько сторонних приложений и веб-сайтов, которые предоставляют аналитические данные и стратегические советы для игроков Rfactor 2.
4. **Sim Racing Telemetry:** Это мобильное приложение, которое работает с различными гоночными симуляторами, включая Assetto Corsa Competizione. Оно предоставляет возможность анализировать технические данные и улучшать навыки вождения.

Каждое из этих приложений имеет свои особенности и преимущества, и выбор зависит от индивидуальных потребностей и предпочтений игрока. Некоторые из них могут быть бесплатными, в то время как другие требуют платной подписки или оплаты за дополнительные функции.

Сравнение аналогов представлено в таблице 1.

Таблица 1 – Сравнение аналогов

	iRacing	Rfactor 2	Sim Racing Telemetry	Приложение, разработанное в рамках курсового проекта
Бесплатная сеть	-	-	+	+
Поддержка русского языка	-	+	-	+
Отсутствие рекламы	+	+	+	+
Удобство использования	+	-	+	+
Отсутствие дополнительных платных функций	+	-	+	+

1.3. Выводы по разделу

В данном разделе была проанализирована предметная область, произведено обоснование актуальности темы, осуществлен обзор аналогов, таких как IRacing, Rfactor 2, Sim Racing Telemetry.

2. Постановка расширенного технического задания

В данном разделе представлено расширенное техническое задание.

В настоящее время набирают популярность автосимуляторы, одним из которых является официальная игра гоночной организации FIA GT Assetto Corsa Competizione.

Разрабатываемое приложение облегчит процесс освоения новичков в спортивном автосимуляторе Assetto Corsa Competizione. Оно включает в себя топливный калькулятор и помощник по настройке автомобилей, а также позволяет отслеживать текущие турниры. Аналогов данного приложения не много. Их функционал разный и некоторые их них распространяются на платной основе.

Глоссарий представлен в таблице 2.

Таблица 2 – Глоссарий

Термин	Определение
Рискованная стратегия	Стратегия на гонку, при которой запас топлива немного (на 1-2 литра) превышает суммарный ожидаемый расход.
Безопасная стратегия	Стратегия на гонку, при которой топливо заливаю в бак с запасом (на 2-3 круга больше, чем ожидается).
TC	Противобуксовочная система - электрогидравлическая система автомобиля, предназначенная для предотвращения потери сцепления колёс с дорогой посредством контроля за буксованием ведущих колёс.
ABS	Антиблокировочная система — система, предотвращающая блокировку колёс транспортного средства при торможении. Основное предназначение системы — сохранение устойчивости и управляемости автомобиля при торможении.

Assetto Corsa Competizione	Гоночная игра, представляющая собой официально лицензированный гоночный симулятор гоночной ассоциацией FIA GT, а также платформой для киберспорта.
Схождение колес	Это сумма углов между плоскостью, которая проходит через центры колес и продольной осью автомобиля.
Развал колес	Угол наклона колес по отношению к перпендикулярной плоскости (в обычном случае — к дороге)
Кастер	Угол, образованный вертикалью и проекцией оси поворота колеса на продольную плоскость автомобиля.
Стабилизатор подвески	устройство в подвеске автомобиля, служащее для уменьшения боковых кренов в поворотах.
Мощность тормозов	Усилие, прикладываемое тормозами при торможении.
Баланс тормозов	Распределение усилия торможения между передней и задней осью автомобиля.
Заднее крыло	Ограждающая конструкция, которая прикрывает колесо транспортного средства, защищает его, а также водителя от грязи и отлетающих камней.
Стабильность поведения автомобиля	Поведение автомобиля меняется не критично по мере отпускания или добавлении газа.
Нестабильность поведения автомобиля	Поведение автомобиля сильно меняется по мере отпускания или добавлении газа.

Диаграмма точек зрения представлена на рисунке 1.

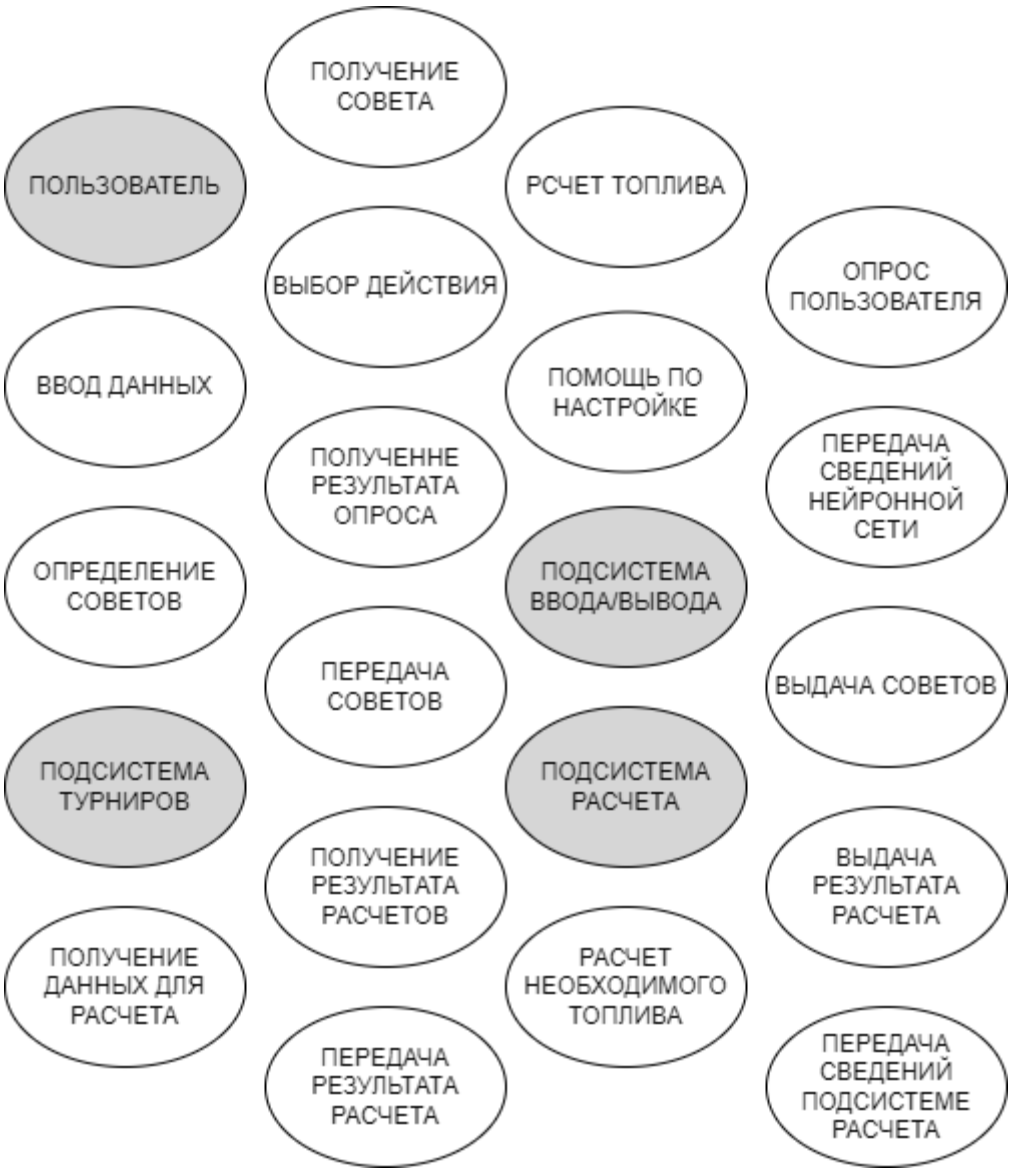

Рисунок 1 – Диаграмма точек зрения

Диаграмма иерархии точек зрения представлена на рисунке 2.

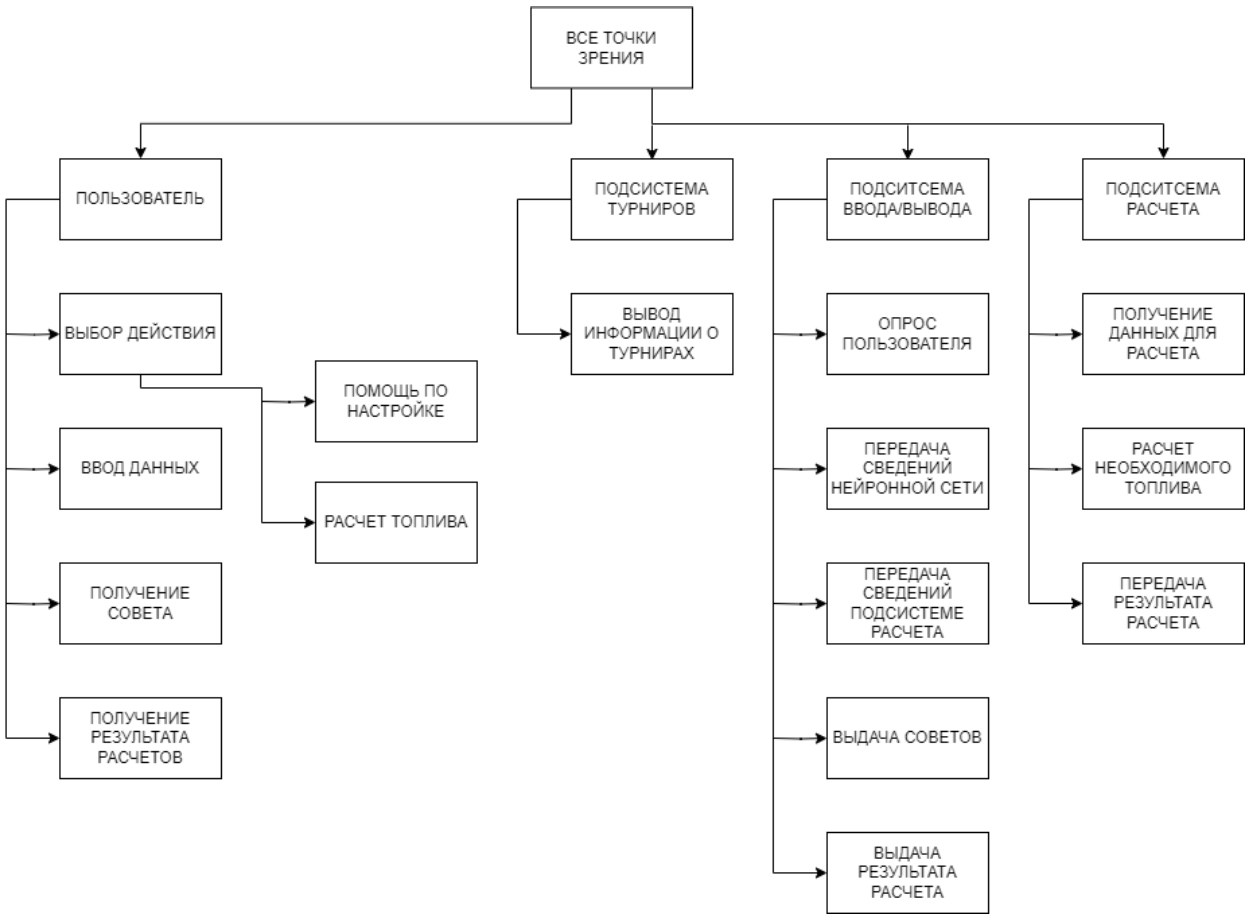

Рисунок 2 – Диаграмма иерархии точек зрения

2.1 Общие положения

2.1.1 Полное наименование

Полное наименование: Интерактивный помощник для игры Assetto Corsa Competizione. Условное обозначение: помощник ACC Helper.

2.1.2 Перечень нормативно-технических документов

При разработке мобильного приложения и создании проектно-эксплуатационной документации Исполнитель должен руководствоваться требованиями следующих нормативных документов:

- ГОСТ 19.201-78. ТЕХНИЧЕСКОЕ ЗАДАНИЕ. ТРЕБОВАНИЯ К СОДЕРЖАНИЮ И ОФОРМЛЕНИЮ;

2.1.3 Определения, обозначения и сокращения

Определения и сокращения представлены в таблице 3.

Таблица 3 – Определения

N	Сокращение	Расшифровка
1	МП	Мобильное приложение
2	ACC	Assetto Corsa Competizione

2.2 Назначение и цели создания системы

2.2.1 Назначение приложения

«ACC Helper» предназначено для помощи новичку путем исполнения следующих функций:

- Расчет топлива для стратегии на гонку
- Помощь в настройке автомобиля
- Отслеживание текущих турниров

2.2.2 Цели создания приложения

Основной целью создания «ACC Helper» является облегчение процесса освоения новичком автосимулятора Assetto Corsa Competizione.

2.3 Характеристика объекта автоматизации

Объектом автоматизации является процесс расчета топлива для гонки. Процесс расчета включает в себя:

- Расчет безопасной стратегии
- Расчет рискованной стратегии
- Расчет вышеуказанных стратегий с учетом и без учета прогревочного круга

2.4 Требования к приложению

2.4.1 Требования к структуре и функционированию приложения

2.4.1.1 Перечень подсистем, их назначение и основные характеристики

- подсистема расчета

Служит для определения безопасной и рискованной стратегий на гонку. Расчет стратегий происходит по формулам, представленным в п. 2.4.3.1. Данные (время круга, расход на один круг, время заезда) для расчета получает от пользователя через интерфейс из подсистемы ввода и вывода.

- подсистема помощи

Служит для формирования и выдачи советов по настройке автомобиля. Совет формируется на основе данных, полученных из опроса пользователя через интерфейс из подсистемы ввода и вывода. Опрос состоит из вопросов, которые описывают поведение автомобиля. Отвечая на них, пользователь будет получать советы о том, как именно изменить параметры (давление в шинах, схождение колес, развал колес, кастер, TC, ABS, карты двигателей, стабилизатор подвески, мощность тормозов, баланс тормозов, коэф. рулевого управления, ход сжатия амортизаторов, разжатие амортизаторов ход, высота, задней и передней подвески, сплиттер, заднее крыло, передние и задние воздухозаборники тормозов) автомобиля.

- подсистема ввода и вывода данных

Служит для считывания данных, необходимых для расчета, и опроса пользователя. Для расчета стратегий подсистема получает от пользователя время круга, расход на один круг, время заезда. А для

определения советов должен быть предусмотрен опрос, состоящий из вопросов, описывающих поведение автомобиля. Передает полученные данные в подсистемы расчета и помощи. Кроме того, выводит пользователю результат вычислений, советы по изменению тех или иных настроек автомобиля, полученные из подсистем помощи и расчетов, а также сводки по турнирным таблицам, полученные из подсистемы турниров.

- подсистема турниров

Служит для получения информации о текущих турнирах. Информацию о турнирах получает с официального сайта YoklmnRacing. Результатом является наборы информации по каждому турниру, которые в дальнейшем выводится через подсистему ввода и вывода.

2.4.1.2 Показатели назначения

Интерактивный помощник «ACC Helper» предназначен для людей, которые только начинают играть в автосимулятор Assetto Corsa Competizione. Процесс осваивания в игре достаточно сложный из-за многочисленных настроек автомобиля, о влиянии которых на поведение автомобиля в самой игре не сказано. Данной приложение облегчит этот процесс, выполняя следующие функции:

- расчет топлива для безопасной и рискованной стратегии на гонку с учетом и без учета прогревочного круга;
- помощь в настройке автомобиля
- предоставление информации о текущих турнирах

2.4.1.3 Требования к эргономике и тех. эстетике

Взаимодействие пользователей с прикладным программным обеспечением, входящим в состав системы должно осуществляться посредством визуального графического интерфейса (GUI). Интерфейс системы должен быть понятным и удобным, не должен быть перегружен

графическими элементами и должен обеспечивать быстрое отображение экранных форм. Навигационные элементы должны быть выполнены в удобной для пользователя форме. Интерфейс должен соответствовать современным эргономическим требованиям и обеспечивать удобный доступ к основным функциям и операциям системы.

Система должна обеспечивать корректную обработку аварийных ситуаций, вызванных неверными действиями пользователей, неверным форматом или недопустимыми значениями входных данных. В указанных случаях система должна выдавать пользователю соответствующие сообщения, после чего возвращаться в рабочее состояние, предшествовавшее неверной (недопустимой) команде или некорректному вводу данных.

Экранные формы должны проектироваться с учетом требований унификации: все экранные формы пользовательского интерфейса должны быть выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации.

2.4.2 Требования к функциям

- Подсистема помощи:
- На основе полученных данных строится математическая модель, которая учитывает множество настроек автомобиля:
    - давление в шинах
    - схождение колес
    - развал колес
    - кастер
    - ТС
    - ABS
    - карты двигателей
    - стабилизатор подвески
    - мощность тормозов
    - баланс тормозов

- коэф. рулевого управления
- ход сжатия амортизаторов
- ход «быстрого» сжатия амортизаторов
- разжатие амортизаторов
- ход «быстрого» разжатия
- высота задней и передней подвески
- сплиттер
- заднее крыло
- передние и задние воздухозаборники тормозов

Исходя из полученных данных происходит определение совета по настройке автомобиля.

- Подсистема расчетов:

Должна осуществлять расчет необходимого для гоночной сессии топлива, основываясь на входных данных, полученных из подсистемы ввода и вывода: время круга, продолжительность заезда, расхода топлива на круг. Расчет проводиться для рискованной и безопасной стратегии, учитывая при необходимости прогревочный круг (у пользователя должна быть возможность выбрать учитывать прогревочный круг или нет).

- Подсистема ввода и вывода данных:

Должна осуществлять считывание введенных пользователем данных, проверять их корректность. Далее передавать данные в подсистему расчетов для определения стратегий. Кроме того, данная подсистема должна опрашивать пользователя о поведении автомобиля и передавать полученные данные в подсистему помощи. Опрос происходит в формате вопрос-ответ. Также она должна выводить информацию о турнирах в виде таблицы результатов. После произведения всех расчетов она должна вывести пользователю информацию о стратегиях и советы по настройке.

- Подсистема турниров

Должна осуществлять сбор информации о текущих турнирах и предоставлять ее в подсистему ввода и вывода данных.

2.4.3 Требования к видам обеспечения

2.4.3.1 Требования к математическому обеспечению системы

Математические методы:

- метод определения стратегий на гонку:

Входные данные: время круга (t), продолжительность заезда (T), расход топлива на круг(l).

Выходные данные: запас топлива (L) для рискованной и безопасной стратегий

Рискованная стратегия, без учета прогревочного круга:

$L = (T/t) * l + 2$

Рискованная стратегия, с учетом прогревочного круга:

$L = (T/t) * l + 2 + l$

Безопасная стратегия, без учета прогревочного круга:

$L = (T/t) * l + 6$

Безопасная стратегия, с учетом прогревочного круга:

$L = (T/t) * l + 6 + l$

2.4.3.2 Требования к техническому обеспечению

Требования к техническим характеристикам смартфона пользователя:

- Объем оперативной памяти – 1 Гб
- Дисковая подсистема – 4 Гб
- Операционная система – Android 4 и выше

2.5 Состав и содержание работ по созданию системы

Разрабатываемое приложение будет поставляться со следующей программной документацией:

- Руководство пользователя, которое включает в себя описание всех возможностей приложения.
- Исходный код с документацией.

2.6 Выводы по разделу

Исходя из вышесказанного, были определены функции, которые необходимы для выполнения цели, а также проанализированы требования по программному и аппаратному обеспечению.

3. Разработка структуры программы

В данном разделе представлен процесс разработки структуры программы, принципа её работы и составных частей с использованием структурного подхода проектирования.

3.1 Описание структуры программы

Необходимо сформировать наиболее точное описание разрабатываемого программного обеспечения. Для этого было принято решение о рассмотрении функциональной диаграммы верхнего уровня.

В данном случае в качестве отображения взаимосвязей была выбрана нотация IDEF0. В качестве входных данных время круга, расход топлива, время заезда, результат опроса, выбор действия пользователем. В качестве субъекта выступает пользователь и вычислительная машина. Управление задается алгоритмами расчета и определения советов, а также алгоритмом сбора данных о турнирах. К выходным данным будут относиться советы, информация о стратегиях на гонку и о турнирах.

Контекстная диаграмма IDEF0 представлена на рисунке 3.

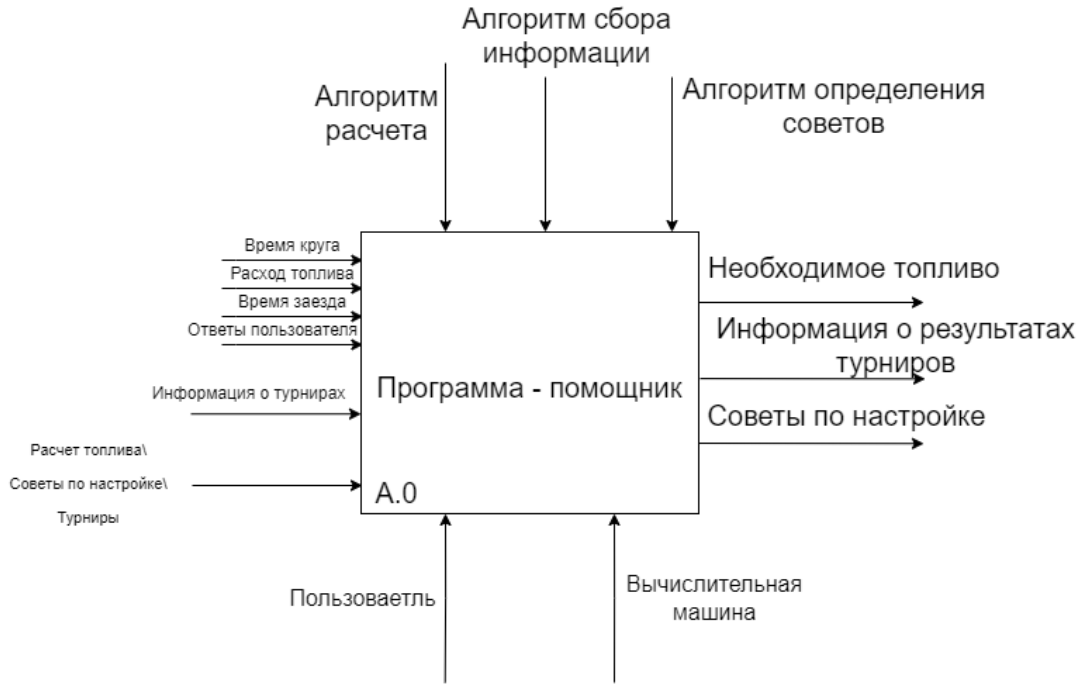

Рисунок 3 – Контекстная диаграмма IDEF0

Детализирующая функциональная диаграмма более подробно раскрывает функциональную диаграмму верхнего уровня: описывает взаимодействия и связи процессов, происходящих внутри системы. На ней можно увидеть, какие процессы взаимосвязаны и что между ними общего.

На детализирующей функциональной диаграмме показаны следующие этапы:

- Выбор действия. Входными данными является выбор пользователем действия, выполняемого программой. Выходным данными является информация для запуска одной из подсистем.
- Расчет топлива. Входными данными является информация о заезде (время круга, расход топлива, время заезда). Выходными данными будут 2 стратегии на гонку.
- Определение советов. Входными данными является результат опроса пользователя. Выходными данными будут советы по настройке автомобиля.
- Информация о турнирах. Входными данными будет старт запуска подсистемы, а выходными данными являются таблицы с результатами текущих турниров.

Декомпозиция контекстной диаграммы представлена на рисунке 4.

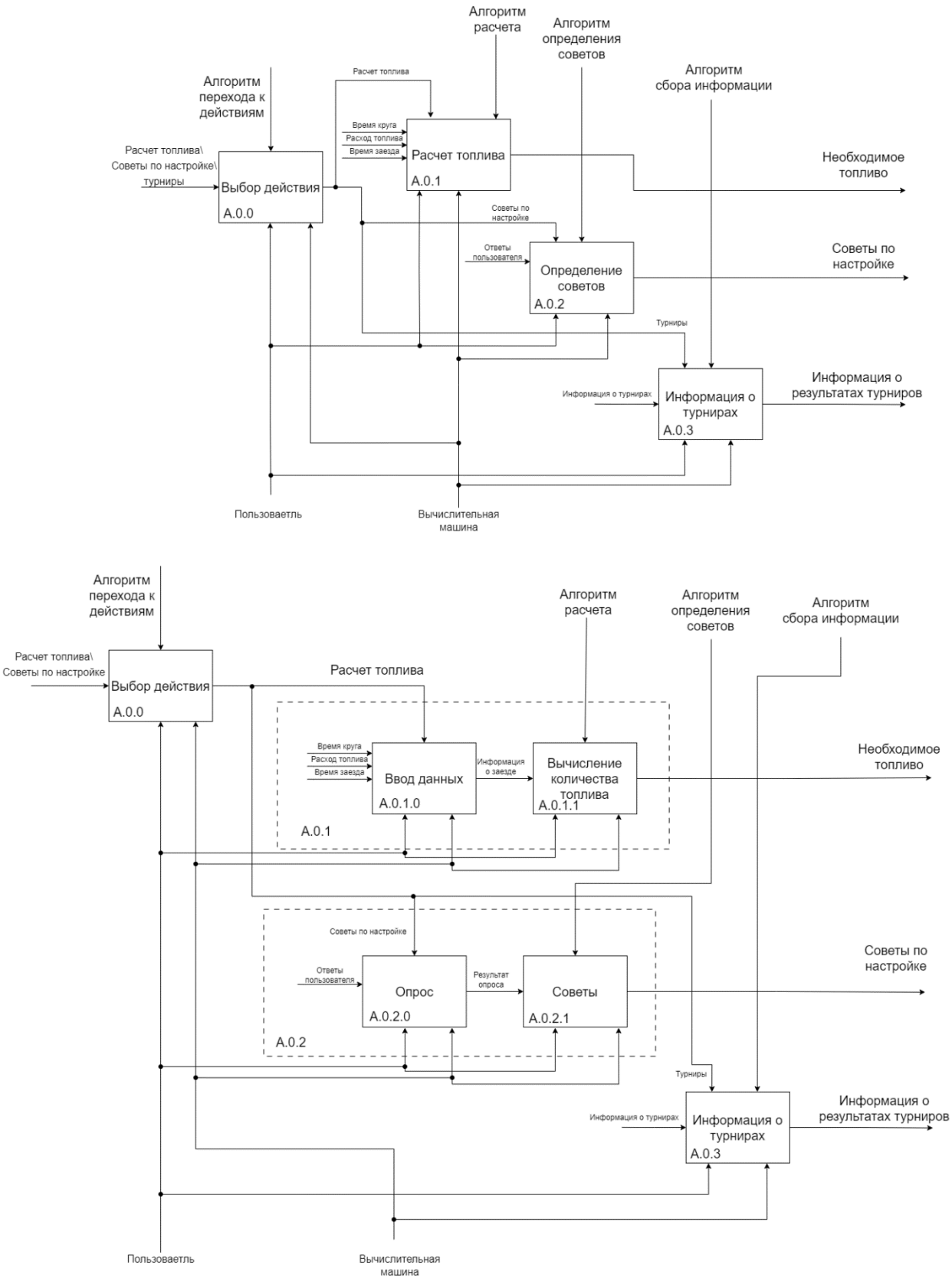

Рисунок 4 - Декомпозиция контекстной диаграммы IDEF0

Для описания динамического поведения объекта в ходе его существования рассмотрим диаграмму переходов состояний. Она наиболее точно выражает как последовательность состояний, испытываемых объектом, событий, вызывающих переходы между состояниями, и действий, сопровождающих переход состояний, происходят в системе. В любом разрабатываемом ПО есть исходное состояние, с которого начинается работа программы. Наиболее подробно описаны действия и события, к которым они приводят. После завершения какого-либо события оно переходит в блок «Ожидание», в котором происходит ожидание дальнейших действий пользователя.

Диаграмма переходов состояний представлена на рисунке 5.

Рисунок 5 – Диаграмма переходов состояний

3.2 Описание структуры базы данных

Хранение информации о турнирах производится в базе данных.

В структуре БД находятся следующие сущности:

- Чемпионат;
- Чарт;
- Команда;
- Этап;
- Водитель;

Атрибуты для каждой сущности представлены на ER-диаграмме. Данная диаграмма представлена на рисунке 6.

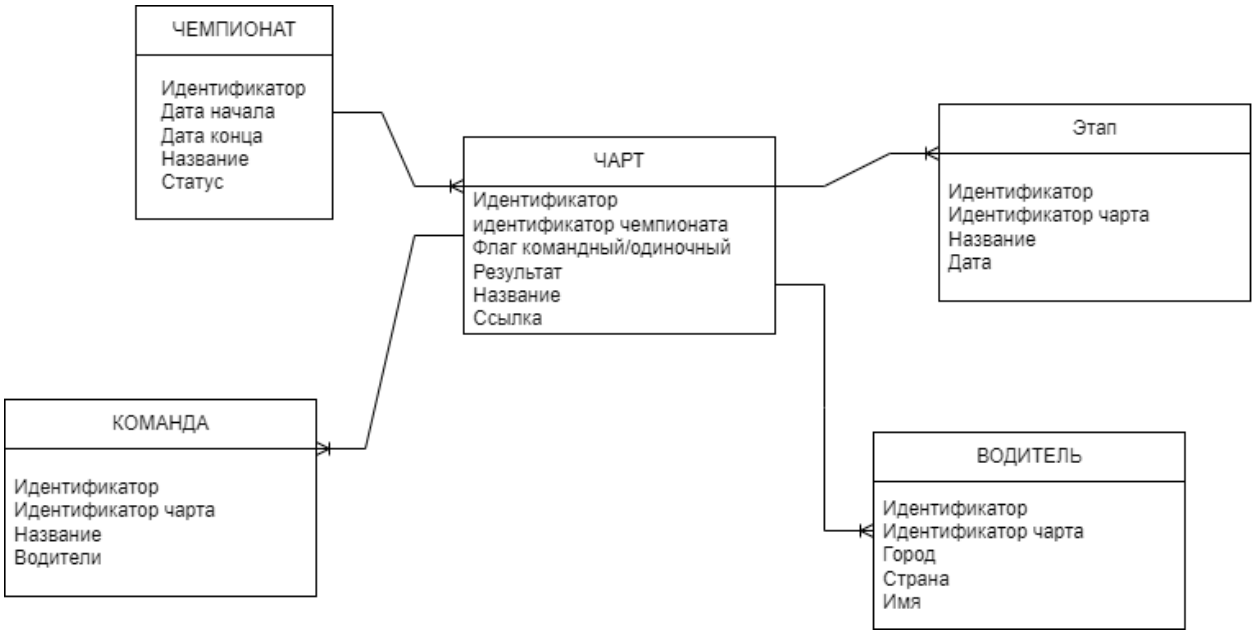

Рисунок 6 – ERD-диаграмма

3.3 Разработка алгоритмов

В данном разделе будут рассмотрены алгоритмы функционирования разрабатываемого приложения.

3.3.1 Алгоритм расчета топлива

Алгоритм расчета топлива предназначен для определения нужного количества топлива, исходя из данных о гонке, полученных от пользователя.

Граф-схема алгоритма представлена на рисунке 7.

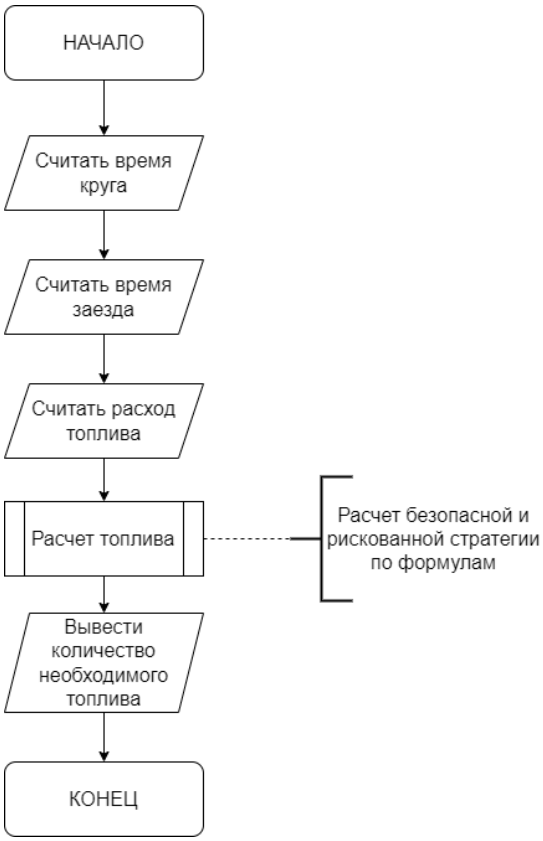

Рисунок 7 – Алгоритм расчета топлива

3.3.2 Алгоритм определения советов

Алгоритм определения советов предназначен для вывода советов по настройке автомобиля на основе полученной от пользователя информации о поведении автомобиля на треке.

Граф-схема алгоритма представлена на рисунке 8.

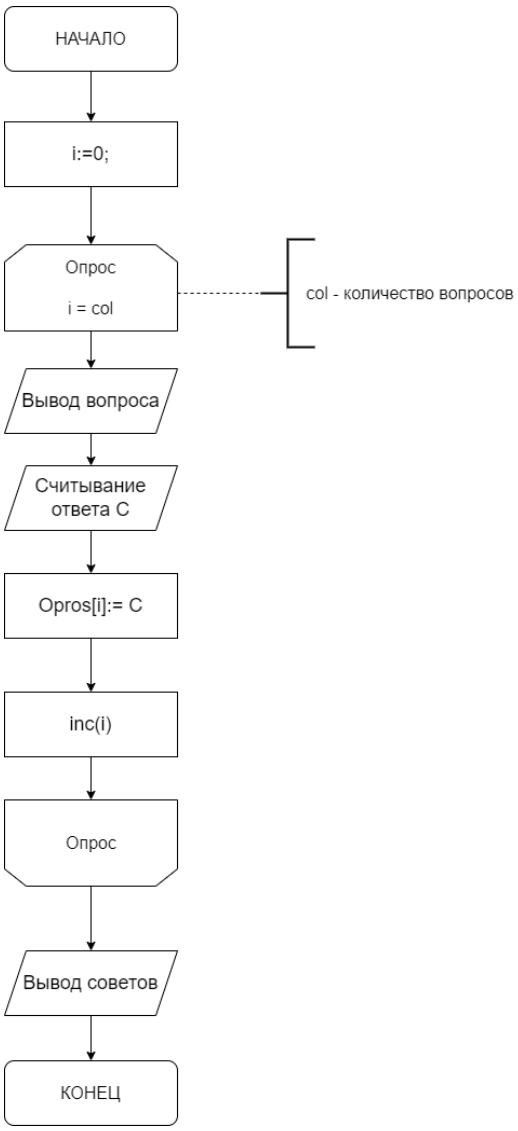

Рисунок 8 – Алгоритм определения советов

3.4 Выводы по разделу

В данном разделе была разработана структура ПО, была произведена работа с базой данных и составлены алгоритмы функционирования.

4 Программная реализация

В данном разделе рассмотрен подход к разработке программного обеспечения, использование средств для разработки, а также разработка интерфейса.

4.1 Выбор инструментов разработки

Программная реализация ACC Helper включает в себя две основные составляющие: frontend и backend. Frontend - это часть приложения, с которой пользователь взаимодействует напрямую. Она отвечает за отображение информации и интерактивные элементы, которые видит пользователь на экране мобильного устройства. Backend - это внутренняя часть приложения, которая обрабатывает запросы от frontend, взаимодействует с базой данных и обеспечивает логику работы приложения.

Frontend (Java, Android Studio):

В разработке frontend ACC Helper используется язык программирования Java и среда разработки Android Studio. Java - это универсальный и мощный язык, который часто используется для разработки мобильных приложений под Android. Android Studio предоставляет удобное и мощное интегрированное средство разработки для создания интерфейса пользователя, обработки событий и взаимодействия с пользователем.

Frontend реализует следующие функции:

- **Отображение данных:** Отображение рекомендаций по стратегии, информации о текущих турнирах, комментариев пользователей и статистики.
- **Интерактивные элементы:** Реализация интерактивных элементов, таких как кнопки, формы для ввода данных, списки и диаграммы для визуализации информации.
- **Обработка пользовательских действий:** Обработка событий, таких как нажатия кнопок, ввод текста и жесты, для взаимодействия пользователя с приложением.

- **Связь с Backend:** Отправка запросов на backend для получения данных и обновления информации на экране.

Backend (Java):

Backend разрабатывается на языке программирования Java. Для доступа к БД используется Spring Data JPA - часть Spring Data проекта, облегчающая доступ к базам данных через JPA. Он представляет собой уровень абстракции над JPA, упрощая и ускоряя разработку.

Backend выполняет следующие функции:

- **Обработка запросов от Frontend:** Принимает запросы от frontend, анализирует их и взаимодействует с базой данных для получения или обновления данных.
- **Управление данными:** Выполняет операции чтения, записи, обновления и удаления данных в базе данных PostgreSQL.
- **Бизнес-логика:** Реализует бизнес-логику приложения, включая анализ данных о предыдущих гонках и генерацию рекомендаций по стратегиям.
- **Связь с БД:** Spring Data JPA позволяет создавать репозитории для взаимодействия с базой данных. Репозитории - это интерфейсы, которые расширяют **JpaRepository** или другие репозитории Spring Data. Они автоматически предоставляют реализацию методов для основных операций CRUD (Create, Read, Update, Delete) над сущностями базы данных.
- **Веб приложение для парсинга:** проект для парсинга данных о турнирах был разработан с помощью Spring Framework - это популярный фреймворк для разработки приложений на языке программирования Java. Он предоставляет обширный набор инструментов и функциональности для упрощения разработки, тестирования и развертывания приложений.

4.2 Разработка интерфейса

Любое приложение начинается с главной страницы. Макет формы начальной страницы представлен на рисунке 9.

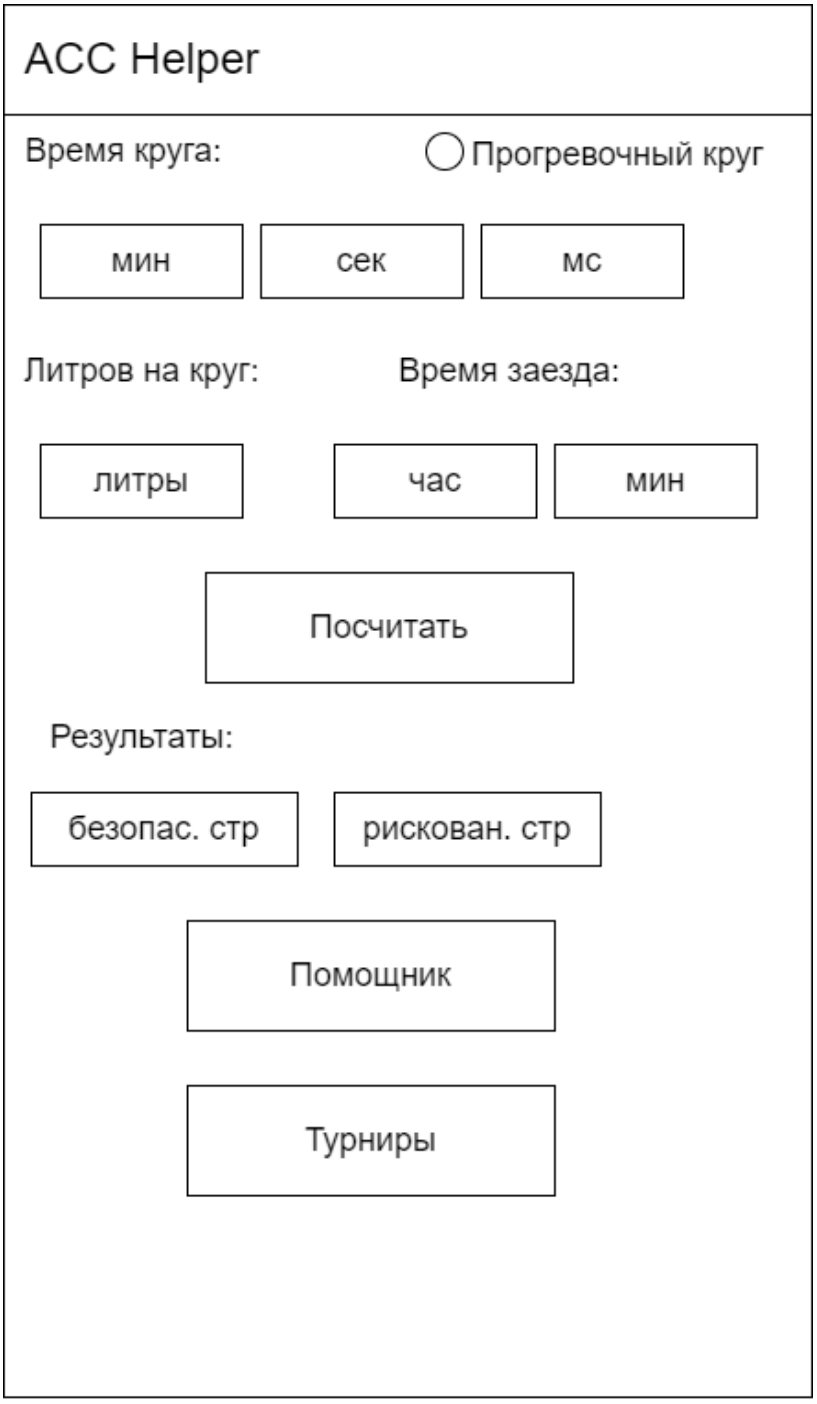

Рисунок 9 – Макет формы начальной страницы

На главном экране присутствуют поля для ввода данных о заезде. после ввода и нажатия кнопки «Посчитать» будет произведен расчет стратегии.

С главной страницы можно совершить переход на экраны определения советов и информации о турнирах.

Макеты форм помощника и сведений о турнирах представлены на рисунках 10 и 11 соответственно.

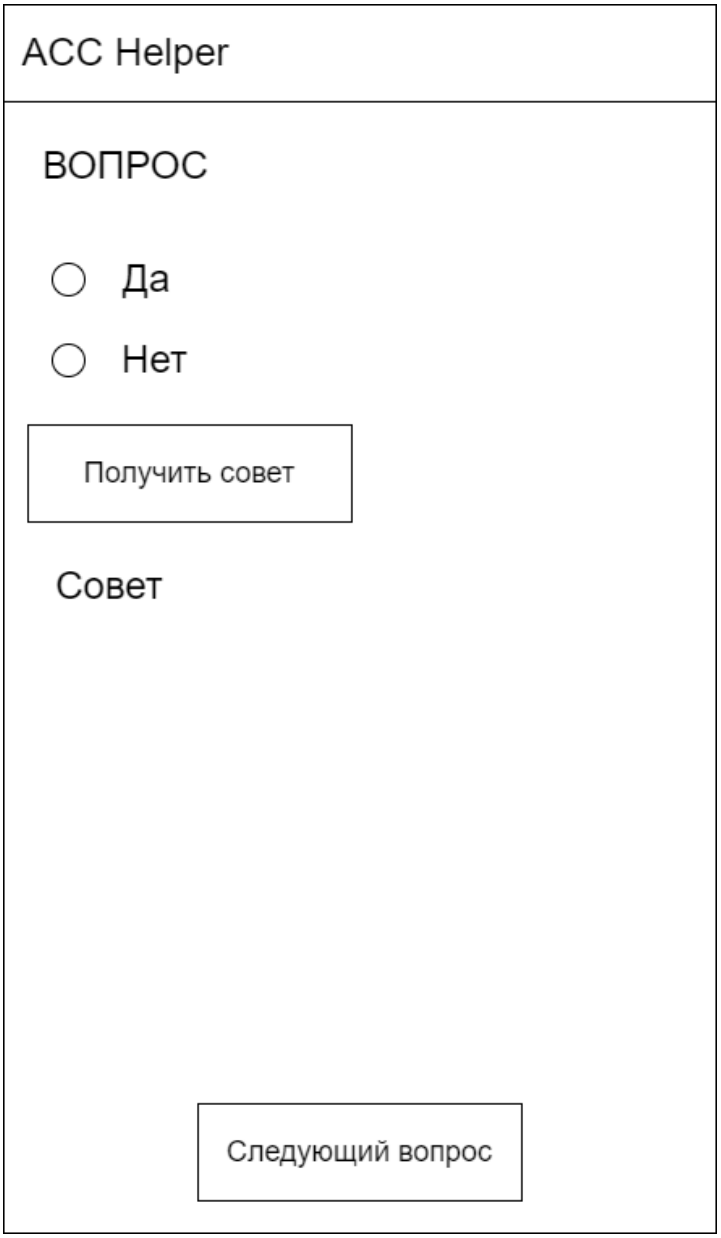

Рисунок 10 – Макет формы помощника

На экране помощника пользователь должен ответить на вопрос и нажать кнопку «Получить совет». После этого на экране появится совет по

настройке автомобиля. Переход к следующему вопросу происходит по нажатию кнопки внизу экрана.

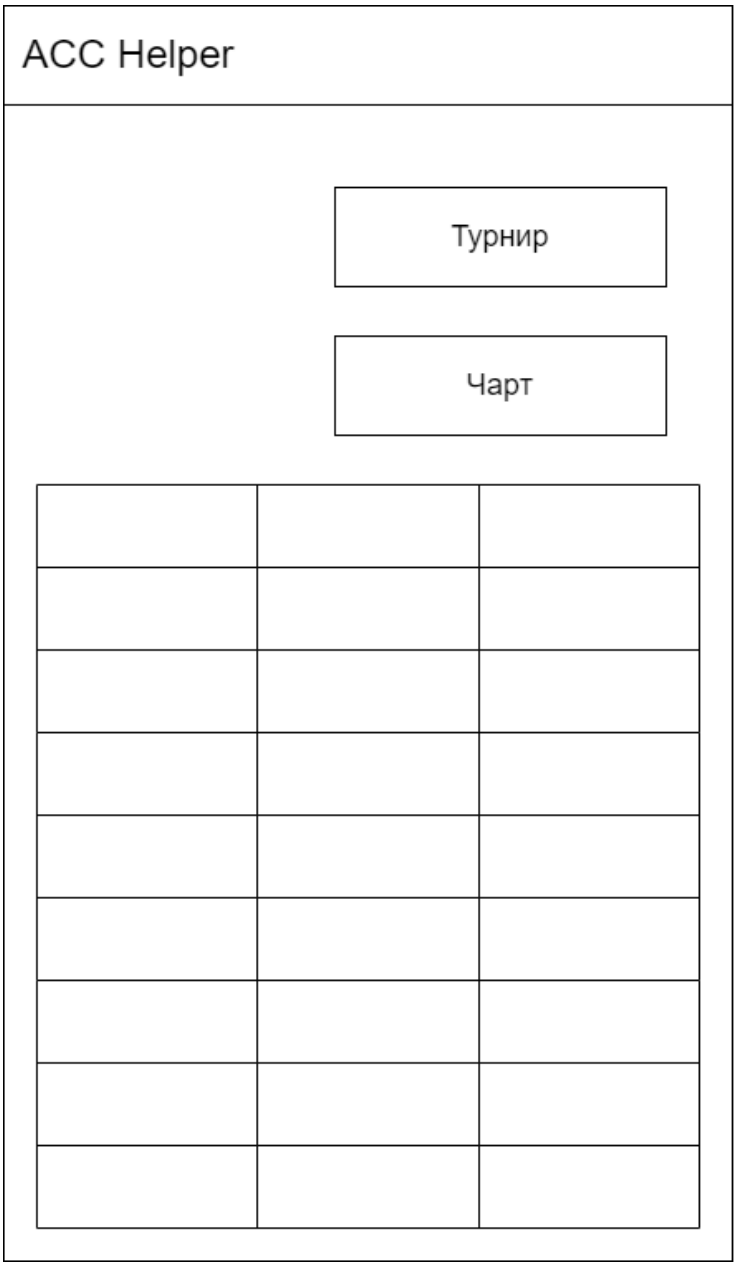

Рисунок 11 – Макет формы экрана с турнирами

На экране с турнирами есть 2 выпадающих списка с турнирами и чартами. Пользователь может выбрать любой турнир и любой чарт, после чего в таблице появится информация о результатах.

4.3 Разработка функционала

4.3.1 Расчет стратегий

Данная функция имеет не сложный функционал. Суть заключается в том, чтобы по готовым формулам, используя данные, которые введет пользователь, посчитать количество необходимого топлива на гонку, то есть определить стратегию.

4.3.2 Советы по настройке

Данная функция также достаточно простая. Для ее реализации был описан массив вопросов для опроса пользователя, фрагмент которого представлен на рисунке 12.

```
private String[] questions = {
        "Ощущаете ли вы недостаточную поворачиваемость при входе в поворот?",
        "Ощущаете ли вы избыточную поворачиваемость при входе в поворот?",
```

Рисунок 12 – Фрагмент массива вопросов для опроса

Далее был описан массив советов по каждому вопросу, фрагмент которого представлен на рисунке 13

```
private String[] adviceOptions = {
        "Уменьшить дорожный просвет спереди на 1 мм.\n" +
                "Уменьшить заднее антикрыло на 1 пункт.\n" +
                "Увеличить задний дорожный просвет на 2 мм.\n" +
                "Увеличить свободный ход передней подвески до упора в ограничители на 1 пункт.\n"
                "Уменьшить жесткость ограничителей хода передней подвески на 1 пункт.\n" +
                "Уменьшить баланс тормозов (сместить в сторону задней оси).",
```

Рисунок 13 – Фрагмент массива советов

Исходя из ответов пользователя на вопросы, ему будут выдаваться советы по настройке.

4.3.3 Разработка приложения для парсинга

4.3.3.1 Описание сущностей

Все сущности БД описаны напрямую в коде с помощью Java Persistence API (JPA). Этот класс использует аннотации JPA для маппинга Java-объектов на таблицы в базе данных. JPA предоставляет способ взаимодействия с базой данных через объектно-ориентированный подход.

Также при разработке использовался Lombok. Он упрощает разработку, автоматически генерируя методы, такие как геттеры, сеттеры, конструкторы и т. д. В классе используются аннотации Lombok (например, **@Builder**, **@AllArgsConstructor**, **@NoArgsConstructor**, **@Getter**, **@Setter**, **@ToString**), чтобы автоматически сгенерировать стандартные методы класса, что делает код более чистым и компактным. Пример использования аннотаций Lombok приведен на рисунке 14.

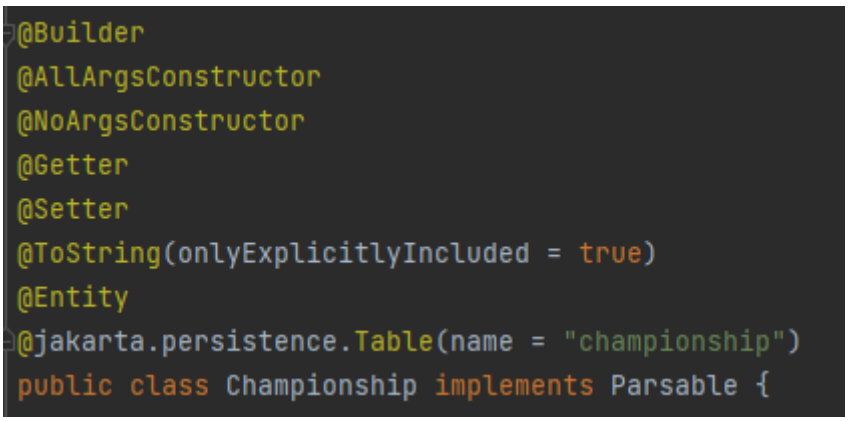

Рисунок 14 – Пример аннотаций

Описание сущностей состоит из объявления полей таблицы (рисунок 15) и объявления связей (рисунок 16).

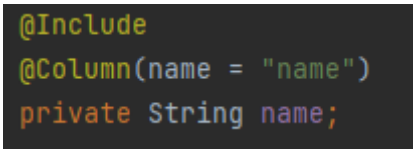

Рисунок 15 – Пример объявления поля таблицы

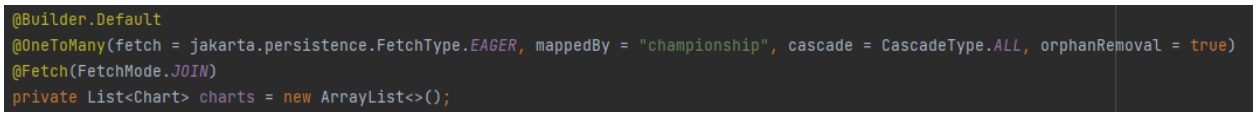

Рисунок 16 – Пример объявления связи

1. Сущность «Чемпионат» содержит следующие поля:
    - идентификатор
    - дата начала
    - дата конца
    - название
    - статус

Имеет связи типа ОДИН-КО-МНОГИМ с сущностью «Чарт».

2. Сущность «Чарт» содержит следующие поля:
    - идентификатор
    - идентификатор чемпионата
    - флаг командный/одиночный
    - результат
    - название
    - ссылка

Имеет связи типа ОДИН-КО-МНОГИМ с сущностями «Команда», «Этап», «Водитель».

3. Сущность «Команда» содержит следующие поля:
    - Идентификатор
    - Идентификатор чарта
    - название
    - водители
4. Сущность «Этап» содержит следующие поля:
    - Идентификатор
    - Идентификатор чарта
    - название
    - дата
5. Сущность «Водитель» содержит следующие поля:
    - Идентификатор
    - Идентификатор чарта
    - Город
    - Страна
    - Имя

Сущность «Чарт» обладает особенностью. Один ее экземпляр имеет связи с экземплярами сущностей «Команда» и «Водитель». Одно из ее полей (флаг командный\одиночный) говорит о том, какая из двух таблиц будет

заполняться. То есть если чарт командный, то флаг будет «1», тогда будет заполняться таблица команды. Если же флаг «0», то чарт одиночный, и заполняться будет таблица водителя.

4.3.3.2 Разработка парсинг-сервисов

Код сервисов представляет собой службу в рамках Spring приложения. О том, что класс является сервисом и должен быть управляемым контейнером Spring, говорит аннотация Spring «@**Service**».

Также используется аннотация Lombok «@**RequiredArgsConstructor**», которая генерирует с final параметрами, для всех полей, помеченных как «final».

Был разработан контроллер **ChampionshipController** для управления данными о чемпионатах. Этот класс является контроллером веб-приложения и обрабатывает HTTP-запросы, связанные с операциями над объектами чемпионата.

1. **@RestController** - Эта аннотация указывает, что класс является контроллером REST API, и он будет обрабатывать входящие HTTP-запросы и возвращать данные в формате JSON.
2. **@RequestMapping("/championships")** - Эта аннотация указывает, что все обработчики запросов в этом контроллере будут обрабатывать URL, начинающиеся с **/championships**.
3. **@Autowired** - Эта аннотация используется для внедрения зависимости **ChampionshipRepository** в контроллер. Это означает, что **ChampionshipController** будет использовать репозиторий для доступа к данным о чемпионатах.
4. **@GetMapping("/")** - Этот метод обрабатывает GET-запросы на **/championships/** и возвращает список всех чемпионатов из репозитория.
5. **@GetMapping("/{Id}")** - Этот метод обрабатывает GET-запросы на **/championships/{Id}**, где **{Id}** - это идентификатор конкретного

чемпионата. Метод пытается найти чемпионат по указанному идентификатору в репозитории. Если чемпионат не найден, возвращается пустой объект чемпионата с id 0 и internalId 0.

Таким образом, этот контроллер предоставляет интерфейс для получения списка всех чемпионатов и получения конкретного чемпионата по его идентификатору через REST API.

Был разработан интерфейс **ParsingService**, определяющий метод **parse()**, который должен быть реализован всеми классами (описаны ниже), выполняющими парсинг веб-страницы и создание объектов, реализующих интерфейс **Parsable**.

Метод parse(Element card):

- Принимает веб-элемент **card**, представляющий информацию на веб-странице, которую необходимо распарсить.
- Возвращает объект, реализующий интерфейс **Parsable**, который содержит данные, извлеченные из веб-элемента.

Этот интерфейс создан для обеспечения абстракции в процессе парсинга веб-страницы. Различные сервисы могут реализовывать этот интерфейс, чтобы обрабатывать разные виды данных и создавать соответствующие объекты, реализующие интерфейс **Parsable**.

Класс **SuperParserService:**

Этот сервис периодически загружает веб-страницу, извлекает информацию о чемпионатах и сохраняет ее в базе данных с использованием Spring Data JPA.

1. **Поля:**
    - **private final ChampionshipService championshipService;**: Ссылка на сервис для обработки данных о чемпионатах.

- **private final ChampionshipRepository championshipRepository;**: Репозиторий для сохранения данных о чемпионатах в базе данных.
- **private List<Element> elementList;**: Список элементов, содержащих информацию о чемпионатах на веб-странице.
- **private final String url = "https://yoklmnracing.ru/championships";**: URL-адрес веб-страницы, которую парсит сервис.

2. **Метод parsing():**
    - Этот метод выполняется после инициализации бина (PostConstruct) и парсит HTML-страницу по указанному URL. Он использует библиотеку Jsoup для выполнения HTTP-запроса и анализа HTML-страницы. Результат парсинга сохраняется в **elementList**.
3. **Метод parseElement():**
    - Этот метод запускается периодически с использованием аннотации **@Scheduled(fixedRate = 300000)**, что означает, что он будет выполняться каждые 300000 миллисекунд (или 5 минут).
    - Он проверяет, пуст ли **elementList**. Если да, то вызывает метод **parsing()** для получения новых данных. Если **elementList** не пуст, он берет первый элемент, парсит информацию о чемпионате, сохраняет ее в базу данных через **championshipService** и удаляет элемент из **elementList**.
4. **Статический метод getId(Element element):**
    - Этот метод получает элемент и возвращает идентификатор чемпионата из этого элемента. Он ищет элемент **<h1>** с классом "card-title.float-start a", извлекает из него атрибут "href" и удаляет все нечисловые символы, возвращая идентификатор в виде строки.

Класс **ChampionshipService:**

Этот сервис ответственен за разбор информации о чемпионатах и их графиках на веб-странице, создание объектов **Championship** и **Chart**, и сохранение их в базе данных через **ChampionshipRepository**.

1. **Поля:**
    - **private final ChampionshipRepository championshipRepository;**: Репозиторий для доступа к сущности **Championship** в базе данных.
    - **private final ChartService chartService;**: Сервис для обработки данных о графиках чемпионатов.
2. **Метод parse(Element card):**
    - Этот метод реализует интерфейс **ParsingService** и принимает в качестве аргумента элемент **card**, представляющий информацию о чемпионате на веб-странице.
    - Метод разбирает элемент **card**, извлекает данные о чемпионате, такие как **internalId**, **name**, **status**, **beginDate**, **endDate** и связанные с ними графики (**charts**).
3. **Разбор данных о чемпионате:**
    - **internalId**: Идентификатор чемпионата, извлекаемый из атрибута "href" элемента **<link rel="canonical">**.
    - **id**: Идентификатор чемпионата в базе данных. Если чемпионат уже существует, используется его идентификатор из базы данных.
    - **name**, **status**: Различные атрибуты чемпионата, извлекаемые из соответствующих элементов веб-страницы.
    - **beginDate** и **endDate**: Даты начала и окончания чемпионата, извлекаемые из элемента на веб-странице и преобразуемые в строки.

- **charts**: Список графиков чемпионата, получаемых с использованием **chartService.parse()**.

4. **Связывание данных:**
    - После создания объекта **Championship**, метод устанавливает связь между этим чемпионатом и его графиками, устанавливая поле **championship** в каждом объекте **Chart**.

Класс **ChartService:**

Этот сервис отвечает за разбор таблицы чемпионата на веб-странице, создание объекта **Chart** и связанных с ним объектов **Stage**, **Racer** и **Team**, а затем сохранение их в базе данных.

1. **Поля:**
    - **private final StageService stageService;**: Сервис для обработки данных о стадиях чемпионата.
    - **private final TeamService teamService;**: Сервис для обработки данных о командах участников чемпионата.
    - **private final RacerService racerService;**: Сервис для обработки данных о гонщиках участников чемпионата.
2. **Метод parse(Element card):**
    - Этот метод реализует интерфейс **ParsingService** и принимает в качестве аргумента элемент **card**, представляющий информацию о таблице чемпионата на веб-странице.
    - Метод разбирает элемент **card**, извлекает данные о таблице чемпионата, такие как **title**, **url**, **isTeam**, **stages**, **result**, а также **racers** и **teams**.
    - Метод также связывает объекты **Chart** с соответствующими объектами **Stage**, **Racer** и **Team**.
3. **Разбор данных о таблице чемпионата:**

- **title**: Заголовок таблицы чемпионата, извлекаемый из элемента на веб-странице. Если таблица является командной, заголовок устанавливается в "Личный".
- **url**: URL-адрес страницы, на которой расположена таблица чемпионата.
- **isTeam**: Флаг, указывающий, является ли таблица командной. Если **tier-select** не существует, таблица считается личной, в противном случае флаг устанавливается в true, если таблица командная.
- **stages**: Список стадий чемпионата, получаемых с использованием **stageService.parse()**.
- **result**: Результаты участников чемпионата, извлекаемые из последнего столбца в таблице.
- **racers**: Список гонщиков, участвующих в чемпионате, получаемых с использованием **racerService.parse()**.
- **teams**: Список команд, участвующих в чемпионате, получаемых с использованием **teamService.parse()**.

Класс **RacerService:**

Этот сервис отвечает за парсинг данных о гонщиках и создание объектов **Racer**, которые могут быть сохранены в базе данных или использованы в других частях приложения.

Метод parse(Element card):

- Этот метод реализует интерфейс **ParsingService** и принимает веб-элемент **card**, содержащий информацию о гонщике.
- Метод извлекает данные о гонщике из различных элементов на веб-странице, таких как имя (**name**), город (**city**), страна (**country**) и URL-адрес профиля (**url**).
- Создает объект **Racer** с полученными данными и возвращает его.

Класс **StageService:**

Этот сервис парсит информацию о соревновательном этапе (Stage) из элемента **card**, который представляет собой веб-элемент с данными о соревновательном этапе.

Метод parse(Element card):

- Этот метод реализует интерфейс **ParsingService** и принимает веб-элемент **card**, содержащий информацию о соревновательном этапе.
- Метод извлекает данные о соревновательном этапе из различных элементов на веб-странице, таких как заголовок (**title**), внутренний идентификатор (**internalId**) и даты начала и окончания этапа (**date**).
- Создает объект **Stage** с полученными данными и возвращает его.

Класс **TeamService:**

Этот сервис парсит информацию о команде из элемента **card**, который представляет собой веб-элемент с данными о команде.

Метод parse(Element card):

- Этот метод реализует интерфейс **ParsingService** и принимает веб-элемент **card**, содержащий информацию о команде.
- Метод извлекает данные о команде из различных элементов на веб-странице, таких как имя команды (**name**) и список участников (**racers**), где каждый участник представлен как отдельная строка.
- Создает объект **Team** с полученными данными и возвращает его.

Таким образом, **SuperParserService** является основным управляющим компонентом, который инициирует процесс парсинга данных с веб-сайта. Далее, **ChampionshipService** управляет парсингом данных о чемпионатах и их связанных объектах, таких как таблицы результатов (**Chart**). Для каждой таблицы результатов используются отдельные сервисы для парсинга данных о этапах, участниках и командах. Все полученные объекты сохраняются в

базе данных через соответствующие репозитории. В результате, вся система представляет собой механизм парсинга, обработки и хранения данных о чемпионатах, участниках, командах и соревновательных этапах с веб-сайта.

Заключение

В результате выполнения курсового проекта было разработано мобильное приложение ACC Helper, которое успешно реализует поставленные в техническом задании задачи, предоставляя геймерам уникальный опыт в мире виртуальных автоспортивных соревнований.

Приложение ACC Helper предоставляет пользователям возможность получать рекомендации по стратегиям в гоночных симуляторах, следить за текущими турнирами, обмениваться опытом и взаимодействовать внутри сообщества. Благодаря функционалу приложения, игроки могут улучшать свои навыки вождения, анализировать результаты и принимать более информированные решения на гоночной трассе.

В дальнейших планах развития приложения внедрение новых возможностей, включая создание групп для более тесного взаимодействия пользователей и добавление видеоуроков, что позволит начинающим гонщикам получать дополнительные знания и навыки в мире автосимуляторов.

Таким образом, ACC Helper представляет собой важный шаг вперед в обогащении игрового опыта геймеров и создании активного и информированного сообщества в мире виртуальных автоспортивных соревнований.

Список литературы

1. Best Practices for Mobile App User Engagement [Электронный ресурс]. – Режим доступа: https://www.smashingmagazine.com
2. Java Programming for Android Developers [Электронный ресурс]. – Режим доступа: https://www.udemy.com
3. РУКОВОДСТВО ПО НАСТРОЙКЕ АВТОМОБИЛЕЙ В ASSETTO CORSA COMPETIZIONE [Электронный ресурс]. – Режим доступа: https://assettocorsacompetizione.ru/setup-guide
4. Docker Documentation [Электронный ресурс]. – Режим доступа: https://docs.docker.com

Приложение А

(обязательное)

Схема алгоритмов

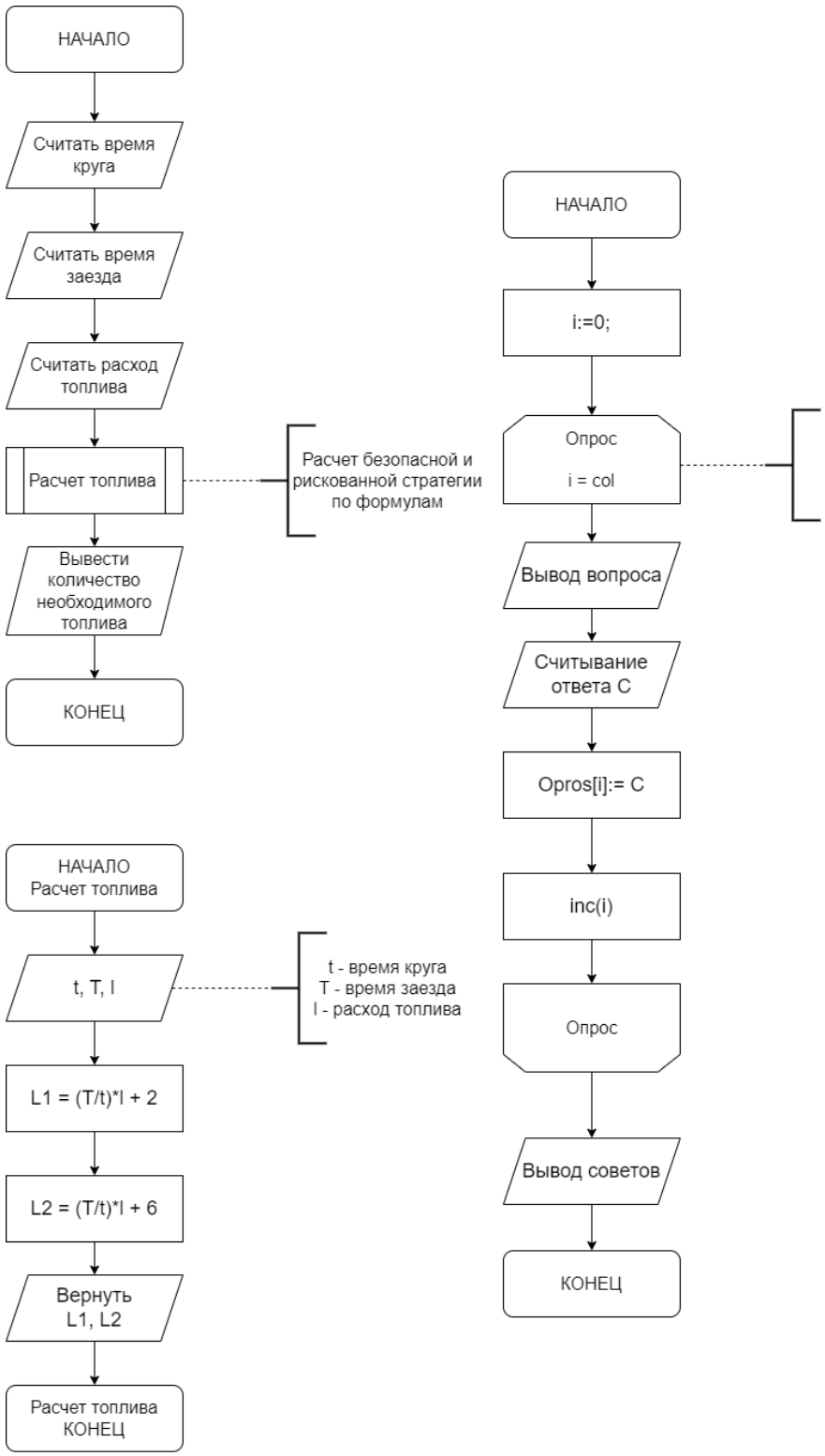

Рисунок 17 – Схемы алгоритмов

Приложение Б

(Обязательное)

Экранные формы

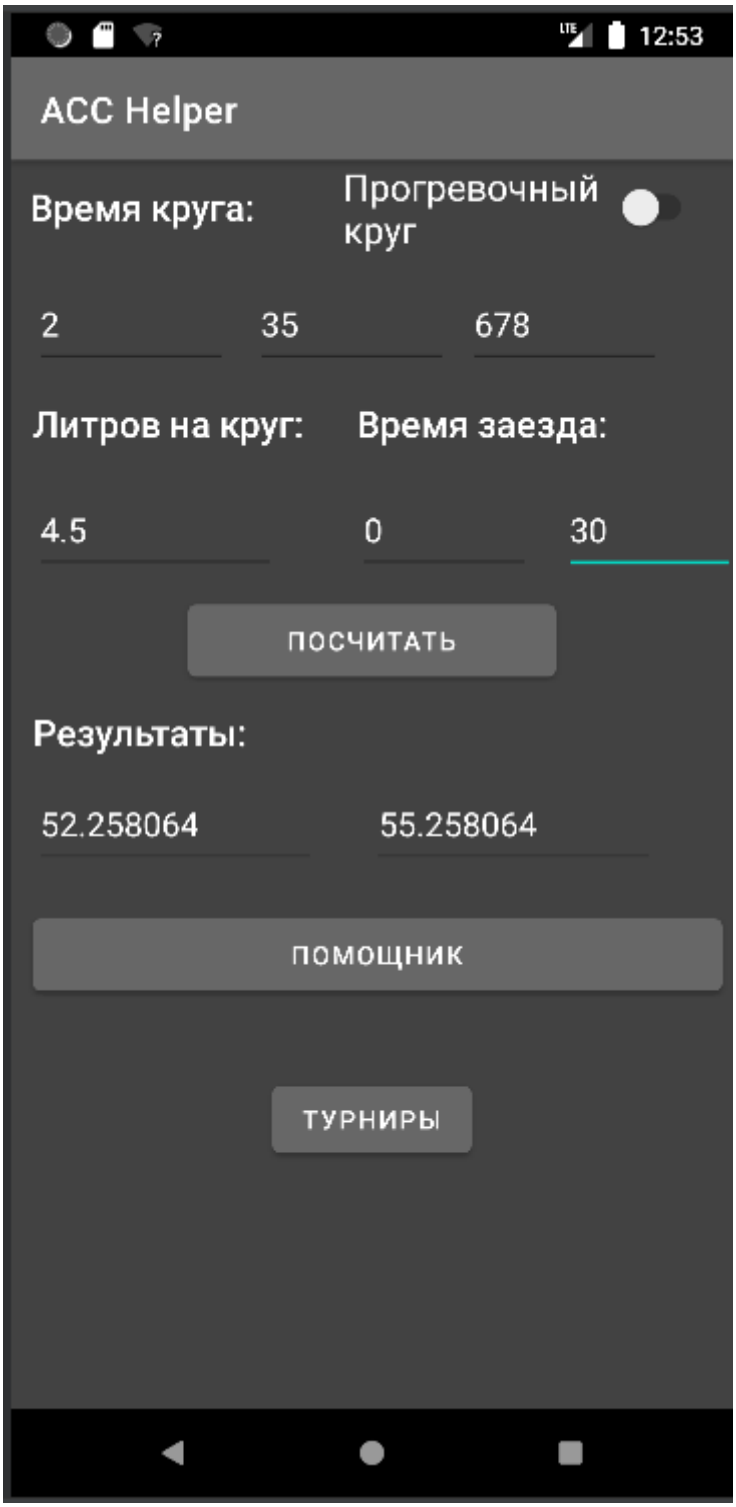

Рисунок 18 – Главный экран

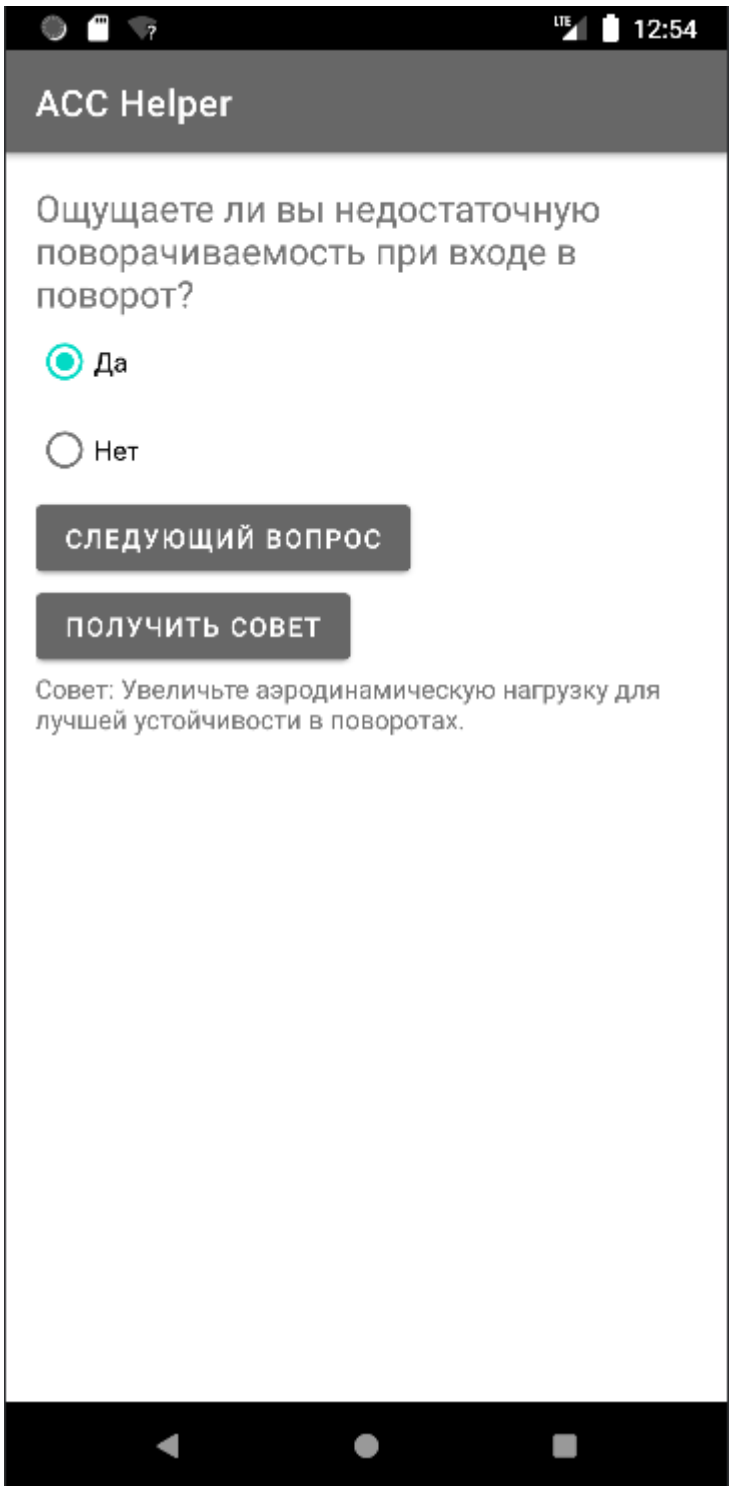

Рисунок 19 – Экран советов

Рисунок 20 – Экран турниров

Приложение В

(Обязательное)

Структура программы

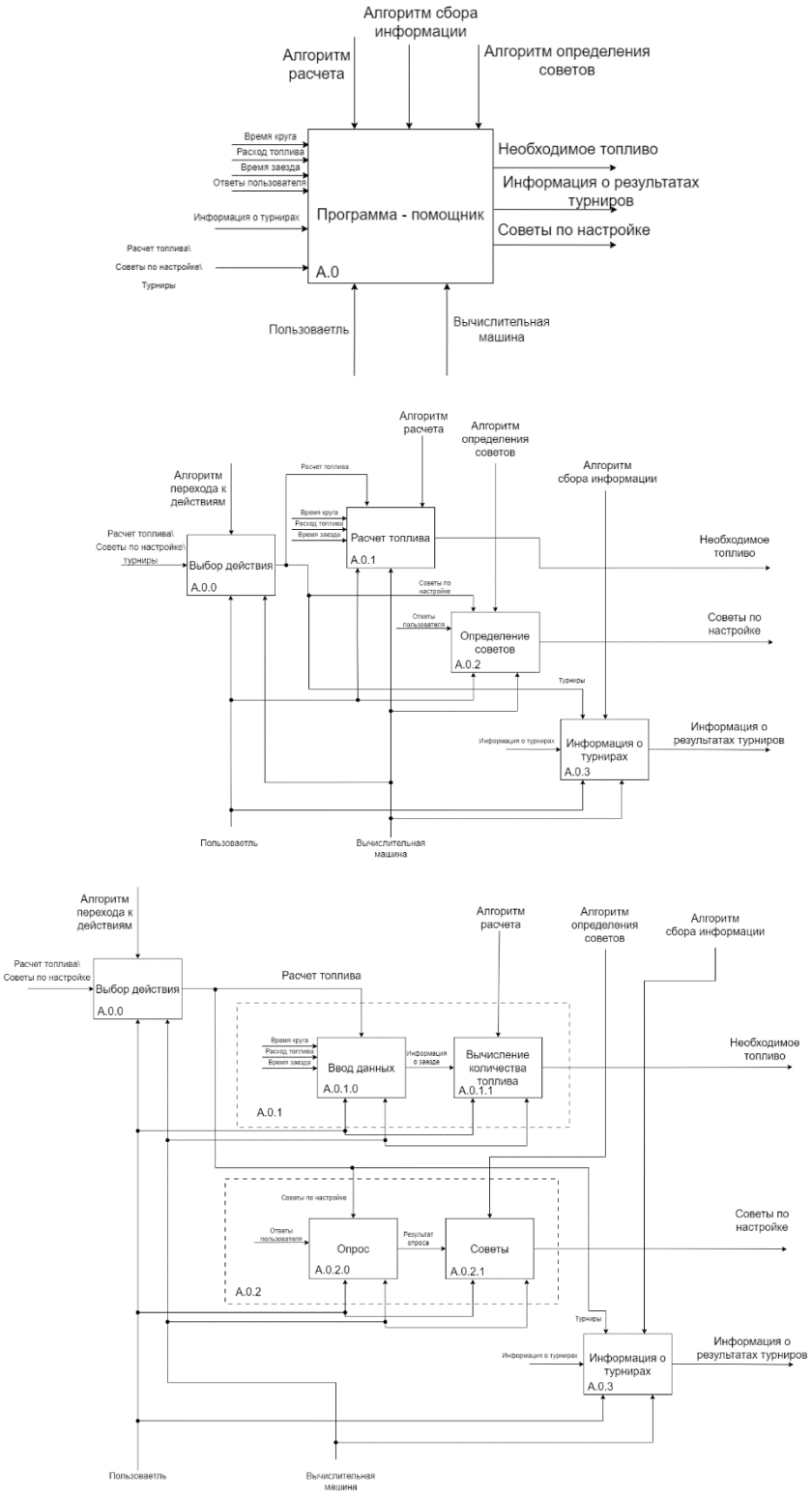

Рисунок 21 – Контекстная и контекстная декомпозированная диаграммы IDEF0

Приложение Г

(Обязательное)

Программный код

```
package com.example.acchelper;

import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.HorizontalScrollView;
import android.widget.ScrollView;
import android.widget.Spinner;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import com.example.acchelper.entity.championship.Championship;
import com.example.acchelper.entity.championship.Chart;
import com.example.acchelper.entity.championship.Row;
import com.example.acchelper.entity.championship.Stage;

import org.springframework.http.converter.json.MappingJackson2HttpMessageConverter;
import org.springframework.web.client.RestTemplate;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class DataAct extends AppCompatActivity {
    TextView textChart;
    TextView textChampionship;
    Spinner spinnerChampionship;
```

```
Spinner spinnerChart;

TableLayout table;
List<Championship> championshipList = new ArrayList<>();

String selectedChampionship;
String selectedChart;

@Override
protected void onCreate(Bundle savedInstanceState) {
    AsyncTask.execute(() -> {
        final String url = "http://192.168.0.101:3000/championships/";
        RestTemplate restTemplate = new RestTemplate();
        List<Championship> championships = new ArrayList<>();
        try {
            for (int i = 1; i < 25; i++) {
                Championship current = null;
                restTemplate.getMessageConverters().add(new MappingJackson2HttpMessageConverter());
                try {
                    current = restTemplate.getForEntity(url + i, Championship.class).getBody();
                } catch (Exception e) {
                }
                if (current != null && current.getId() != 0) {
                    championships.add(current);
                }
            }
        } catch (Exception e) {
            Championship championship = new Championship();
            championship.setName("null");
            Chart chart = new Chart();
            chart.setTitle("null");
            championship.getCharts().add(chart);
            championshipList.add(championship);
        }
        if (championships.size() != 0) {
            championshipList.addAll(championships);
            fillInSpinnerChampionship();
        }
```

```
        });

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_data);

        textChampionship = findViewById(R.id.textChampionship);
        textChart = findViewById(R.id.textChart);
        spinnerChampionship = findViewById(R.id.spinnerChampionship);
        spinnerChart = findViewById(R.id.spinnerChart);
        table = findViewById(R.id.table);



        textChampionship.setText("Выберите чемпионат:");
        textChart.setText("Выберите таблицу:");
        fillInSpinnerChampionship();
        spinnerChampionship.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position,
long id) {
                selectedChampionship = (String) parentView.getItemAtPosition(position);
                List<String> updatedChartNames = getChartNamesForChampionship(selectedChampionship);
                ArrayAdapter<String> updatedChartAdapter = new ArrayAdapter<>(DataAct.this,
android.R.layout.simple_spinner_item, updatedChartNames);

updatedChartAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                spinnerChart.setAdapter(updatedChartAdapter);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parentView) {
                // Обработка случая, когда не выбран ни один чемпионат
            }
        });
        spinnerChart.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position,
long id) {
                selectedChart = (String) parentView.getItemAtPosition(position);
```

```
                updateTable(selectedChampionship, selectedChart);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parentView) {
            }
        });
    }

    private void fillInSpinnerChampionship() {
        runOnUiThread(() -> {
            List<String> championshipNames = new ArrayList<>();
            for (Championship championship : championshipList) {
                championshipNames.add(championship.getName());
            }
            ArrayAdapter<String> championshipAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, championshipNames);

championshipAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            spinnerChampionship.setAdapter(championshipAdapter);
        });
    }


    private List<String> getChartNamesForChampionship(String selectedChampionship) {
        List<String> chartNames = new ArrayList<>();
        for (Championship championship : championshipList) {
            if (Objects.equals(championship.getName(), selectedChampionship)) {
                for (Chart chart : championship.getCharts()) {
                    chartNames.add(chart.getTitle());
                }
                break;
            }
        }
        return chartNames;
    }
    private void updateTable(String selectedChampionship, String selectedChart) {
        table.removeAllViews();
```

```
Chart currentChart = null;
for (Championship championship : championshipList) {
    if (Objects.equals(championship.getName(), selectedChampionship)) {
        for (Chart chart : championship.getCharts()) {
            if (Objects.equals(chart.getTitle(), selectedChart)) {
                currentChart = chart;
                break;
            }
        }
    }
}

if (currentChart != null) {
    TableRow headerRow = new TableRow(this);
    //Next
    int i = 0;
    TextView textViewTable = new TextView(this);
    textViewTable.setText("Таблица");
    i++;
    textViewTable.setPadding(3, 3, 3, 3);
    headerRow.addView(textViewTable);
    //Next
    for (Stage stage : currentChart.getStages()) {
        TextView textViewStage = new TextView(this);
        textViewStage.setText("S" + i);
        if (i % 2 == 1)
            textViewStage.setBackgroundColor(Color.GRAY);
        i++;
        textViewStage.setPadding(3, 3, 3, 3);
        headerRow.addView(textViewStage);
    }
    //Next
    TextView textViewFinalResult = new TextView(this);
    textViewFinalResult.setText("Результаты");
    if (i % 2 == 1)
        textViewFinalResult.setBackgroundColor(Color.GRAY);
    i++;
```

```
textViewFinalResult.setPadding(3, 3, 3, 3);
headerRow.addView(textViewFinalResult);
if (currentChart.getStages().size() % 2 == 0)
    i++;
//Next
table.addView(headerRow);
for (int j = 0; j < currentChart.getResult().size(); j++) {
    TableRow row = new TableRow(this);
    //Next
    String name;
    if (currentChart.getIsTeam()) {
        name = currentChart.getTeams().get(j).getName();
    } else {
        name = currentChart.getRacers().get(j).getName();
    }
    TextView textViewParticipant = new TextView(this);
    textViewParticipant.setText(name);
    textViewParticipant.setPadding(3, 3, 3, 3);
    if (i % 2 == 1)
        textViewParticipant.setBackgroundColor(Color.GRAY);
    i++;
    row.addView(textViewParticipant);
    //Next
    for (String s : currentChart.getRows().get(j).getData()) {
        TextView textViewData = new TextView(this);
        textViewData.setText(s);
        if (i % 2 == 1)
            textViewData.setBackgroundColor(Color.GRAY);
        i++;
        textViewData.setPadding(3, 3, 3, 3);
        row.addView(textViewData);
    }
    //Next
    TextView textViewResult = new TextView(this);
    textViewResult.setText(currentChart.getResult().get(j));
    if (i % 2 == 1)
        textViewResult.setBackgroundColor(Color.GRAY);
```

```
                i++;
                textViewResult.setPadding(3, 3, 3, 3);
                row.addView(textViewResult);
                if (currentChart.getStages().size() % 2 == 0)
                    i++;
                //Next
                table.addView(row);
            }
        }
    }
}
```

```
package com.example.acchelper;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.util.JsonReader;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.example.acchelper.entity.championship.Championship;

import org.springframework.http.ResponseEntity;
import org.springframework.http.client.SimpleClientHttpRequestFactory;
import org.springframework.http.converter.json.MappingJackson2HttpMessageConverter;
import org.springframework.web.client.RestTemplate;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.lang.reflect.ParameterizedType;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

import javax.net.ssl.HttpsURLConnection;



public class MainActivity extends AppCompatActivity {
    // Объявление имен полей для ввода
    EditText Edit_min;
    EditText Edit_sec;
    EditText Edit_litres;
    EditText Edit_RHr;
    EditText Edit_RMin;
    EditText Edit_Result_Safely;
    EditText Edit_Result_Risky;
    Button myButton;
    Button btnGoToSecAct;
    Button btnGoToDataAct;
    @SuppressLint("UseSwitchCompatOrMaterialCode")
    Switch SW_Heat_Lap;



    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

/*      GettingAsyncTask asyncTask = new GettingAsyncTask();
        asyncTask.execute();*/
```

```
// Связь имен с полями на форме
Edit_min = findViewById(R.id.Edit_min);
Edit_sec = findViewById(R.id.Edit_sec);
Edit_litres = findViewById(R.id.Edit_litres);
Edit_RHr = findViewById(R.id.edit_RHr);
Edit_RMin = findViewById(R.id.edit_RMin);
Edit_Result_Safely = findViewById(R.id.Edit_Result_Safely);
Edit_Result_Risky = findViewById(R.id.Edit_Result_Risky);
myButton = findViewById(R.id.Calculate_button);
SW_Heat_Lap = findViewById(R.id.Switch_Heat_Lap);
btnGoToSecAct = findViewById(R.id.btnGoToSecAct);
btnGoToDataAct = findViewById(R.id.btnGoToDataAct);

View.OnClickListener oclBtnGoToSecAct = view -> {
   Intent intent = new Intent(MainActivity.this, SecAct.class);
   startActivity(intent);
};
btnGoToSecAct.setOnClickListener(oclBtnGoToSecAct);


View.OnClickListener oclCalcBtn = view -> {

   if (!Edit_min.getText().toString().trim().equals("") &
!Edit_sec.getText().toString().trim().equals("") &
      !Edit_litres.getText().toString().trim().equals("")
      & !Edit_RHr.getText().toString().trim().equals("") &
!Edit_RMin.getText().toString().trim().equals("")) {

      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
         Edit_Result_Safely.setTextColor(getColor(R.color.white));
         Edit_Result_Risky.setTextColor(getColor(R.color.white));
      }
      Edit_Result_Safely.setText("");
      Edit_Result_Risky.setText("");

      String min = Edit_min.getText().toString();
      String sec = Edit_sec.getText().toString();
      float seconds = Float.parseFloat(sec) / 60;
```

```
        int Minutes = Integer.parseInt(min);
        float TotalLapTime = Minutes + seconds;

        String RHoursStr = Edit_RHr.getText().toString();
        String RMinutesStr = Edit_RMin.getText().toString();

        int RHours = Integer.parseInt(RHoursStr) * 60;

        int RMinutes = Integer.parseInt(RMinutesStr);
        int RaceTime = RHours + RMinutes;

        float LapCount = RaceTime / TotalLapTime;

        String LitresPerLap = Edit_litres.getText().toString();
        float Litres = Float.parseFloat(LitresPerLap);

        float TotalLitres = 0;
        if (!SW_Heat_Lap.isChecked()) TotalLitres = Litres * LapCount;
        else if (SW_Heat_Lap.isChecked()) TotalLitres = Litres * LapCount + Litres;

        Edit_Result_Safely.setText(String.valueOf(TotalLitres));
        Edit_Result_Risky.setText(String.valueOf(TotalLitres + 3));
    } else {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            Edit_Result_Safely.setTextColor(getColor(R.color.red));
            Edit_Result_Risky.setTextColor(getColor(R.color.red));
        }
        Edit_Result_Safely.setText(R.string.Error);
        Edit_Result_Risky.setText(R.string.Error);
    }
};
myButton.setOnClickListener(oclCalcBtn);

View.OnClickListener oclBtnGoToDataAct = view -> {
    Intent intent = new Intent(MainActivity.this, DataAct.class);
    startActivity(intent);
};
```

```
        btnGoToDataAct.setOnClickListener(oclBtnGoToDataAct);
      }



}
package education.AssettoCorsaParser.controllers;

import education.AssettoCorsaParser.entity.championship.Championship;
import education.AssettoCorsaParser.repository.championship.ChampionshipRepository;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/championships")
public class ChampionshipController {

  private final ChampionshipRepository championshipRepository;

  @Autowired
  public ChampionshipController(ChampionshipRepository championshipRepository) {
    this.championshipRepository = championshipRepository;
  }

  @GetMapping("/")
    public List<Championship> getAll(){
    return championshipRepository.findAll();
    }

  @GetMapping("/{Id}")
  public Championship getChampionshipById(@PathVariable Long Id) {
    return
championshipRepository.findById(Id).orElse(Championship.builder().id(0L).internalId(0).build());
  }
```

```
}

package education.AssettoCorsaParser.entity.championship;

import education.AssettoCorsaParser.entity.Parsable;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.Setter;
import lombok.ToString;
import lombok.ToString.Include;
import org.hibernate.annotations.Fetch;
import org.hibernate.annotations.FetchMode;

@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@ToString(onlyExplicitlyIncluded = true)
@Entity
@jakarta.persistence.Table(name = "championship")
public class Championship implements Parsable {

  @Include
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @Include
    @Column(name = "internal_id")
    private Integer internalId;

    @Include
    @Column(name = "name")
    private String name;

    @Include
    @Column(name = "status")
    private String status;

    @Include
    @Column(name = "simulator")
    private String simulator;

    @Include
    @Column(name = "organization")
    private String organization;

    @Include
    @Column(name = "begin_date")
    private String beginDate;

    @Include
    @Column(name = "end_date")
    private String endDate;

    @Builder.Default
    @OneToMany(fetch = jakarta.persistence.FetchType.EAGER, mappedBy = "championship", cascade =
CascadeType.ALL, orphanRemoval = true)
    @Fetch(FetchMode.JOIN)
```

```
  private List<Chart> charts = new ArrayList<>();
}

package education.AssettoCorsaParser.entity.championship;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import education.AssettoCorsaParser.entity.Parsable;
import education.AssettoCorsaParser.entity.participant.Racer;
import education.AssettoCorsaParser.entity.participant.Team;
import jakarta.persistence.CascadeType;
import jakarta.persistence.CollectionTable;
import jakarta.persistence.Column;
import jakarta.persistence.ElementCollection;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import java.util.ArrayList;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.Setter;
import lombok.ToString;
import lombok.ToString.Include;
import org.hibernate.annotations.Fetch;
import org.hibernate.annotations.FetchMode;

@Builder
@AllArgsConstructor
```

```
@NoArgsConstructor
@Getter
@Setter
@ToString(onlyExplicitlyIncluded = true)
@Entity
@Table(name = "chart")
public class Chart implements Parsable {

 @Include
 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(name = "id", nullable = false)
 private Long id;

 @Include
 @Column(name = "title")
 private String title;

 @Include
 @Column(name = "url")
 private String url;

 @Include
 @Column(name = "is_team")
 private Boolean isTeam;

 @Include
 @Builder.Default
 @ElementCollection
 @Column(name = "result")
 @CollectionTable(name = "chart_result", joinColumns = @JoinColumn(name = "owner_id"))
 private List<String> result = new ArrayList<>();

 @JsonIgnore
 @ManyToOne
 @JoinColumn(name = "championship_id")
 @Fetch(FetchMode.JOIN)
```

```
    private Championship championship;

    @Builder.Default
    @OneToMany(fetch = jakarta.persistence.FetchType.EAGER, mappedBy = "chart", cascade = CascadeType.ALL, orphanRemoval = true)
    @Fetch(FetchMode.JOIN)
    private List<Row> rows = new ArrayList<>();

    @Builder.Default
    @OneToMany(fetch = jakarta.persistence.FetchType.EAGER, mappedBy = "chart", cascade = CascadeType.ALL, orphanRemoval = true)
    @Fetch(FetchMode.JOIN)
    private List<Stage> stages = new ArrayList<>();

    @Builder.Default
    @OneToMany(fetch = jakarta.persistence.FetchType.EAGER, mappedBy = "chart", cascade = CascadeType.ALL, orphanRemoval = true)
    @Fetch(FetchMode.JOIN)
    private List<Racer> racers = new ArrayList<>();

    @Builder.Default
    @OneToMany(fetch = jakarta.persistence.FetchType.EAGER, mappedBy = "chart", cascade = CascadeType.ALL, orphanRemoval = true)
    @Fetch(FetchMode.JOIN)
    private List<Team> teams = new ArrayList<>();
}
```

```
package education.AssettoCorsaParser.entity.participant;

import com.fasterxml.jackson.annotation.JsonIgnore;
import education.AssettoCorsaParser.entity.Parsable;
import education.AssettoCorsaParser.entity.championship.Chart;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
```

```
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.Setter;
import lombok.ToString;
import lombok.ToString.Include;

@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@ToString(onlyExplicitlyIncluded = true)
@Entity
@jakarta.persistence.Table(name = "racer")
public class Racer implements Parsable {

 @Include
 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 @Column(name = "id", nullable = false)
 private Long id;

 @Include
 @Column(name = "url")
 private String url;

 @Include
 @Column(name = "name")
 private String name;

 @Include
 @Column(name = "city")
```

```
  private String city;

  @Include
  @Column(name = "country")
  private String country;

  @JsonIgnore
  @ManyToOne
  @JoinColumn(name = "chart_id")
  private Chart chart;
}
```

```
package education.AssettoCorsaParser.service.parser;

import education.AssettoCorsaParser.entity.championship.Championship;
import education.AssettoCorsaParser.repository.championship.ChampionshipRepository;
import education.AssettoCorsaParser.service.ChampionshipService;
import jakarta.annotation.PostConstruct;
import java.io.IOException;
import java.util.List;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
@Slf4j
public class SuperParserService {

  private final ChampionshipService championshipService;
  private final ChampionshipRepository championshipRepository;
```

```
private List<Element> elementList;
private final String url = "https://yoklmnracing.ru/championships";

@PostConstruct
public void parsing() {
  try {
    // Получение HTML-страницы с сайта
    Document baseDoc = Jsoup.connect(url).get();
    // Поиск всех элементов с классом "card mb-3", которые содержат информацию о чемпионатах
    elementList = baseDoc.select("div.card.mb-3");
    // Перебор всех найденных элементов
  } catch (IOException e) {
    e.printStackTrace();
  }
}

@Scheduled(fixedRate = 300000)
private void parseElement() {
  if (elementList.isEmpty()) {
    parsing();
  } else {
    try {
      Element element = elementList.get(0);
      log.atInfo().log("Start parsing " + url + "/" + getId(element));
      Championship championship =
          championshipService.parse(Jsoup.connect(url + "/" + getId(element)).get());
      championshipRepository.save(championship);
      elementList.remove(element);
      log.atInfo().log("Parsing element successfully!");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}

static private String getId(Element element) {
  Element e = element.selectFirst("h1.card-title.float-start a");
```

```
    if (e == null) {
      return "";
    } else {
      return e.attr("href").replaceAll("\\D", "");
    }
  }

}
```

```
package education.AssettoCorsaParser.service;

import education.AssettoCorsaParser.entity.championship.Championship;
import education.AssettoCorsaParser.entity.championship.Chart;
import education.AssettoCorsaParser.repository.championship.ChampionshipRepository;
import jakarta.transaction.Transactional;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Optional;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Element;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
@Slf4j
public class ChampionshipService implements ParsingService {

  private final ChampionshipRepository championshipRepository;
  private final ChartService chartService;
```

```
@Override
public Championship parse(Element card) {
  log.atInfo().log("  Championship parsing");
  int internalId = 0;
  {
    try {
      internalId = Integer.parseInt(
          card.select("link[rel=canonical]").attr("href").replaceAll("\\D", ""));
    } catch (Exception exception) {
      log.atWarn().log(card.baseUri() + " - Championship.InternalId: " + exception.getMessage());
    }
  }
  Long id = null;
  {
    Optional<Championship> optionalChampionship = championshipRepository.findByInternalId(
        internalId);
    if (optionalChampionship.isPresent()) {
      id = optionalChampionship.get().getId();
    }
  }
  String name = card.select("h1.card-title").text();
  String status = card.select("td:has(i.fab.fa-solid.fa-flag-checkered) + td").text();
  String organization = card.select("td:contains(Организатор) + td a").text();
  String simulator = card.select("td:has(i.fab.fa-solid.fa-gamepad)").text();
//   LocalDate beginDate = null;
//   LocalDate endDate = null;
  String beginDate = "";
  String endDate = "";
  {
    Element elementDate = card.select("td:has(i.fab.fa-solid.fa-calendar-days) + td")
        .first();
    if (elementDate != null) {
      String[] dateParts = elementDate.text().trim().split(" - ");
//       DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy",
//           Locale.ENGLISH);
//       beginDate = LocalDate.parse(dateParts[0], formatter);
//       endDate = LocalDate.parse(dateParts[1], formatter);
```

```
    beginDate = dateParts[0];
    endDate = dateParts[1];
   }
  }
  List<Chart> charts = new ArrayList<>();
  {
   Element selectorTables = card.getElementById("tier-select");
   String urlTable =
      "https://yoklmnracing.ru/championships/" + internalId + "?tab=standings";
   if (selectorTables == null) {
    try {
     charts.add(chartService.parse(Jsoup.connect(urlTable).get()));
    } catch (Exception exception) {
     log.atWarn().log(card.baseUri() + " - Championship.OneChart: " + exception.getMessage());
    }
   } else {
    for (Element e : selectorTables.select("option")) {
     try {
      String url = urlTable + "&" + e.attr("value");
      charts.add(chartService.parse(Jsoup.connect(url).get()));
     } catch (Exception exception) {
      log.atWarn().log(card.baseUri() + " - Championship.Chart: " + exception.getMessage());
     }
    }
   }
  }
  Championship championship = Championship.builder()
    .id(id)
    .internalId(internalId)
    .name(name)
    .status(status)
    .organization(organization)
    .simulator(simulator)
    .beginDate(beginDate)
    .endDate(endDate)
    .charts(charts)
    .build();
```

```
    for (Chart chart : charts) {
      chart.setChampionship(championship);
    }
    log.atInfo().log("  " + name + " - Championship was successfully parsed.");
    return championship;
  }

}
```

```
package education.AssettoCorsaParser;

import jakarta.annotation.PostConstruct;
import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication()
@EnableJpaRepositories
@EnableScheduling
@Slf4j
public class AssettoCorsaParserApplication {
	public static void main(String[] args) {
		SpringApplication.run(AssettoCorsaParserApplication.class, args);
	}
}
```