Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет инженерно-экономический
Кафедра экономической информатики
Дисциплина «Основы алгоритмизации и программирования»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсовой работы
Ассистент кафедры ЭИ
_____ А.В.Канаш
___.____.2025

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему:
«РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ПОДДЕРЖКИ ПРОКАТА АВТОМОБИЛЕЙ»

БГУИР КР 6-05-0611-01 005 ПЗ

Выполнил студент группы 478105
Писарик Мирослав Игоревич
_____
(подпись студента)
Курсовая работа представлена на проверку ___.____.2025
_____
(подпись студента)

Минск 2025

РЕФЕРАТ

БГУИР КР 1-40 05 01-02 088 ПЗ
ТЕМА: курсовая работа / [ФИО студента]. – Минск: БГУИР, 2024. – п.з. – ХХ с., рисунков – 0, таблиц – ХХ, источников – ХХ, приложений – ХХ, чертежей (плакатов) – 0 л. Формата А4.

*Объект исследования***:** Автоматизированная система поддержки проката автомобилей.

Предмет исследования: Анализ требований к системе, проектирование и разработка программного средства для автоматизации процесса проката автомобилей с использованием объектно-ориентированного подхода и базы данных SQLite.

*Цель курсовой работы***:** Разработка автоматизированной системы для упрощения и оптимизации процесса проката автомобилей, улучшения пользовательского опыта и обеспечения эффективного управления автопарком.

Методы исследования: Анализ требований, проектирование системы с использованием объектно-ориентированного программирования, реализация программного кода на языке C++ с использованием базы данных SQLite, тестирование функциональности системы.

Результаты работы: Разработана система, позволяющая пользователям регистрироваться, авторизовываться, просматривать и арендовать автомобили с учетом возраста и опыта вождения. Администраторы могут управлять пользователями и автопарком. Программа реализована на языке C++ с использованием SQLite для хранения данных и консольного интерфейса для взаимодействия с пользователем. Программный продукт разработан на языке *C++* с применением *MS Visual Studio 2022*.

Область применения результатов: Система может быть использована в компаниях по прокату автомобилей для автоматизации процессов аренды, управления автопарком и обеспечения безопасности данных пользователей.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы курсовой работы обусловлена растущим спросом на услуги проката автомобилей и необходимостью автоматизации связанных процессов для повышения эффективности и удобства пользователей. Традиционные методы проката автомобилей часто связаны с ручным оформлением документов, длительными процессами и отсутствием оперативной информации о доступных автомобилях. Разработка автоматизированной системы поддержки проката автомобилей позволяет устранить эти недостатки, предоставляя цифровую платформу для аренды и управления автопарком.

Объект исследования: Автоматизированная система поддержки проката автомобилей.

Предмет исследования: Разработка программного средства для автоматизации процесса проката автомобилей с использованием языка C++ и базы данных SQLite.

Цель курсовой работы: Создание программной системы, которая оптимизирует процесс проката автомобилей, обеспечивая удобство для пользователей и администраторов, а также безопасность данных.

Для достижения поставленной цели были определены следующие задачи:

- Изучить требования к системе проката автомобилей;
- Спроектировать архитектуру системы с использованием объектно-ориентированного подхода;
- Реализовать программное средство на языке C++ с применением SQLite;
- Провести тестирование системы для проверки ее функциональности и надежности.

Программа обладает практической значимостью, предоставляя современное решение для автоматизации проката автомобилей, что востребовано как пользователями, так и компаниями, предоставляющими такие услуги.

Записка состоит из четырёх разделов, каждый из которых решает отдельные задачи: исследование процесса учета успеваемости студентов (далее процесса), его анализ, проектирование и разработка программной поддержки системы учета успеваемости студентов.

1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРОЦЕССА ПРОКАТА АВТОМОБИЛЕЙ

1.1 **Описание процесса проката автомобилей**

Процесс проката автомобилей представляет собой комплексную последовательность действий, охватывающую регистрацию пользователей, выбор доступных автомобилей, оформление аренды и управление автопарком, что позволяет организовать взаимодействие между клиентами и сервисом проката. Автоматизированная система, разработанная для этих целей, направлена на обеспечение удобства, эффективности и прозрачности на каждом этапе взаимодействия. Она должна предоставлять пользователям возможность пройти регистрацию и авторизацию, при этом учитывая их возраст и опыт вождения, чтобы гарантировать соответствие требованиям безопасности и правилам проката. После успешной авторизации система отображает перечень автомобилей, доступных для аренды, с возможностью фильтрации по различным критериям, таким как тип автомобиля, ценовой диапазон или технические характеристики, что помогает пользователям быстро найти подходящий вариант. Процесс аренды автомобилей также учитывает индивидуальные условия, такие как стаж вождения клиента, чтобы обеспечить соответствие требованиям для конкретных категорий автомобилей, например, более мощных или премиальных моделей. Для администраторов система предусматривает функции управления, которые позволяют добавлять новые автомобили в автопарк, удалять устаревшие или неисправные машины, а также управлять учетными записями пользователей, включая блокировку тех, кто нарушает правила сервиса, или разблокировку после устранения нарушений.

Основная цель этой системы заключается в упрощении взаимодействия между пользователями и сервисом проката, минимизации временных затрат на оформление аренды и обеспечении полной прозрачности всех этапов процесса, чтобы клиенты могли легко ориентироваться в условиях аренды, а администраторы эффективно управляли ресурсами. Такой подход позволяет сократить ручные операции, снизить вероятность ошибок и повысить уровень удовлетворенности клиентов за счет быстрого и удобного доступа к услугам проката. Кроме того, автоматизация способствует повышению безопасности, так как система может автоматически проверять соответствие пользователя требованиям для аренды определенного автомобиля, исключая возможные риски, связанные с недостаточным опытом вождения или возрастом клиента.

1.2 Классификация информационных систем в сфере проката автомобилей

Информационные системы в сфере проката автомобилей играют важную роль в оптимизации процессов аренды, улучшении взаимодействия с клиентами и повышении эффективности управления автопарком. Они обеспечивают автоматизацию рутинных задач, упрощают доступ к услугам для пользователей и способствуют более прозрачной и управляемой работе компаний, предоставляющих автомобили в аренду. Классификация таких систем может быть проведена на основе различных критериев, таких как функциональность, область применения и тип обрабатываемых данных, что позволяет глубже понять их назначение и особенности внедрения в индустрии проката автомобилей.

Одним из ключевых критериев классификации является функциональность, которая определяет, какие задачи решает система в процессе проката автомобилей. В рамках этого подхода выделяются системы управления автопарком, которые отвечают за отслеживание доступных автомобилей, их технических характеристик, состояния и текущей доступности для аренды, обеспечивая администраторов актуальной информацией для принятия решений. Другим типом являются системы учета пользователей, которые сосредоточены на управлении данными клиентов, включая их регистрационные данные, историю аренды, статус учетной записи и доступ к услугам, что позволяет персонализировать взаимодействие с каждым пользователем и обеспечивать безопасность данных. Также существуют системы аренды, которые автоматизируют процесс выбора и бронирования автомобилей, предоставляя пользователям удобный интерфейс для поиска подходящих машин, оформления аренды и получения подтверждений, что значительно сокращает время на выполнение этих операций.

Еще одним важным критерием классификации является тип обрабатываемых данных. Некоторые системы фокусируются на данных об автомобилях, таких как модель, год выпуска, пробег, стоимость аренды и минимальные требования к водителю, что помогает в управлении автопарком и предоставлении клиентам точной информации. Другие системы ориентированы на пользовательские данные, включая имя, возраст, стаж вождения, предпочтения и историю платежей, что позволяет адаптировать услуги под индивидуальные потребности клиентов. Также существуют системы, которые обрабатывают данные о транзакциях, такие как сроки аренды, стоимость, штрафы и возвраты, обеспечивая прозрачность финансовых операций и упрощая учет доходов компании.

Классификация по области применения позволяет выделить системы, ориентированные на разные аспекты работы прокатной компании. Например, системы управления автопарком и их обслуживанием помогают отслеживать техническое состояние автомобилей, планировать их ремонт и обеспечивать своевременное обновление автопарка, что снижает риски сбоев в работе сервиса. Системы клиентского взаимодействия предоставляют пользователям возможность регистрироваться, авторизовываться, просматривать доступные автомобили и оформлять аренду через удобные интерфейсы, включая веб-сайты и мобильные приложения. Системы аналитики и отчетности собирают данные о деятельности компании, такие как частота аренды, популярность моделей автомобилей и поведение клиентов, предоставляя руководству информацию для стратегического планирования и повышения качества услуг.

Разработанная система объединяет элементы всех упомянутых типов, предоставляя комплексное решение для проката автомобилей, которое охватывает управление автопарком, учет пользователей и автоматизацию процесса аренды. Такой подход позволяет создать универсальную платформу, которая удовлетворяет потребности как клиентов, так и администраторов, обеспечивая удобство, эффективность и прозрачность на всех этапах взаимодействия.

Для иллюстрации применения подобных систем в индустрии проката автомобилей можно привести примеры известных решений, используемых в различных компаниях по всему миру. Например, система RentalMan, популярная в США, представляет собой интегрированное решение для управления автопарком, бронирования и учета клиентов, широко применяемое в крупных прокатных компаниях для оптимизации процессов аренды. Еще одним примером является платформа RentSyst, которая используется в Европе и предоставляет возможности для управления автопарком, автоматизации бронирований и аналитики данных, что помогает компаниям адаптироваться к изменяющимся рыночным условиям. Система Bluebird Auto Rental Systems, также применяемая в США, ориентирована на комплексное управление прокатным бизнесом, включая учет автомобилей, клиентов и финансов, что делает ее популярной среди средних и крупных предприятий. Наконец, платформа TSD Rental, используемая по всему миру, предлагает решения для управления автопарком, бронирования и интеграции с платежными системами, что позволяет компаниям повысить эффективность работы и улучшить клиентский опыт.

Таким образом, классификация информационных систем в сфере проката автомобилей помогает лучше понять их роль, функции и возможности применения в индустрии. Эти системы способствуют автоматизации процессов, улучшению взаимодействия с клиентами, повышению

прозрачности операций и оптимизации управления автопарком. Внедрение таких решений позволяет прокатным компаниям эффективно справляться с растущими запросами клиентов, минимизировать временные затраты и обеспечивать высокое качество обслуживания, что делает их неотъемлемой частью современного рынка аренды автомобилей.

2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОЙ ПОДДЕРЖКИ ПРОЦЕССА ПРОКАТА АВТОМОБИЛЕЙ

2.1 Постановка задачи.

Автоматизированная система поддержки проката автомобилей предназначена для повышения эффективности и удобства управления процессом аренды транспортных средств, оптимизации работы автопарка и улучшения качества обслуживания клиентов. Внедрение такой системы позволяет упростить процесс аренды автомобилей, автоматизировать учет данных о пользователях и транспортных средствах, а также обеспечить надежное управление доступом для различных ролей пользователей. Основное преимущество системы заключается в минимизации ручных операций, снижении вероятности ошибок и предоставлении прозрачного и удобного интерфейса для взаимодействия как клиентов, так и администраторов.

Административный интерфейс системы предоставляет расширенные возможности управления: от добавления и удаления автомобилей в автопарк до контроля учетных записей пользователей, включая их блокировку или разблокировку. Пользователи получают удобный инструментарий для просмотра доступных автомобилей с учетом их возраста и опыта вождения, а также для оформления аренды в несколько шагов. Особое внимание уделено безопасности системы — реализована авторизация с проверкой пароля, хеширование паролей для защиты данных и ограничение доступа на основе ролей пользователей.

Требования к функционалу ПО:

1 Упрощение процесса аренды автомобилей:

- Возможность для пользователей просматривать доступные автомобили через консольный интерфейс с учетом их возраста и опыта вождения.
- Автоматическая проверка соответствия пользователя требованиям автомобиля (возраст и опыт) при аренде.

2 Работа с пользователями:

- Ведение базы данных пользователей с возможностью хранения имени, хешированного пароля, возраста, опыта вождения, рейтинга, статуса блокировки и прав администратора.

3 Управление автопарком:

- Возможность добавления новых автомобилей с указанием типа, модели, базовой цены и минимального возраста аренды.
- Удаление автомобилей из базы данных по модели.

Роли пользователей:
Администратор:

- Управление учетными записями пользователей (блокировка/разблокировка).
- Управление автопарком (добавление и удаление автомобилей).

Пользователь:

- Самостоятельный просмотр доступных автомобилей через интерфейс.
- Оформление аренды автомобиля.

Нефункциональные требования:

- Безопасность: хеширование паролей пользователей, ограничение доступа на основе ролей, защита от некорректного ввода данных.
- Производительность: обеспечение быстрого доступа к данным через SQLite, минимизация задержек при выполнении операций.

2.2 Разработка функциональной модели предметной области в нотации IDEF0.

Применение методики IDF0 при анализе учёта успеваемости студентов позволяет выделить ключевые моменты, взаимосвязи и функциональные блоки информационной системы. Этот подход структурирует информацию о процессе, обеспечивая возможность оптимизации учёта успеваемости с увеличением его эффективности и точности.

Анализ процесса проката автомобилей в нотации IDEF0 представляет собой структурированный подход к оценке методов и процедур, используемых для обеспечения аренды транспортных средств в рамках автоматизированной системы. При анализе процесса проката автомобилей в нотации IDEF0 первоначально определяются основные компоненты и шаги этого процесса. В рамках данной главы будет рассмотрен каждый уровень IDEF0 диаграммы, начиная с общего процесса проката автомобилей, представленного на рисунке 1, и заканчивая детализированными этапами выполнения каждого подпроцесса.

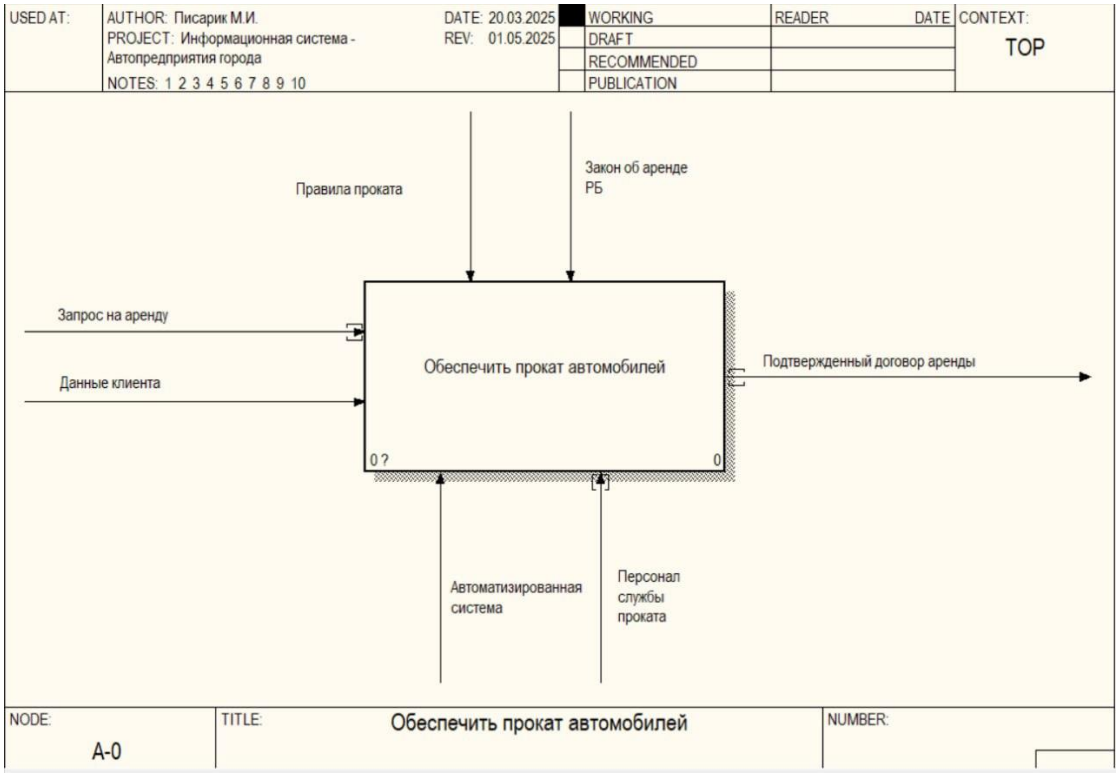

Рисунок 2.1 – Контекстная диаграмма

Диаграмма представляет собой общий обзор процесса проката автомобилей, выполненный в нотации IDEF0. Это контекстный уровень (A-0), который иллюстрирует взаимодействие системы с внешними факторами. Основные компоненты включают входные данные (запросы клиентов на аренду автомобилей), выходные данные (подписанные договоры аренды), управляющие факторы (правила проката, установленные компанией) и механизмы (автоматизированная система). Диаграмма задает основу для более детализированных уровней анализа.

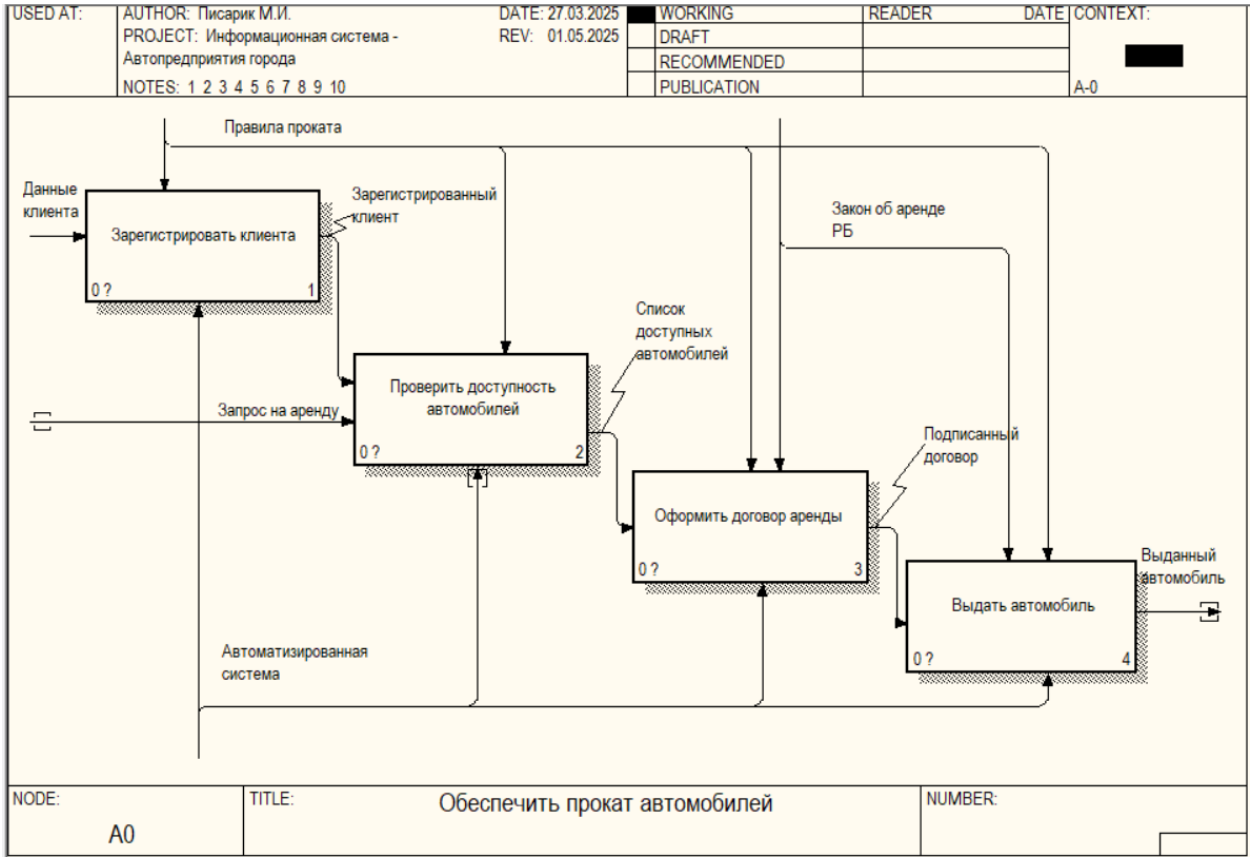

Рисунок 2.2 – блок A0 “Обеспечить прокат автомобилей”

На втором уровне диаграммы IDEF0, изображенного на рисунке 2.2, представлен представлен процесс обеспечения проката автомобилей. Этот уровень дает обзор основных этапов учета и определяет общий порядок выполнения процесса. В блоке A0 выделяются следующие этапы:

1 Зарегистрировать клиента.
2 Проверить доступность автомобилей.
3 Оформить договор аренды.
4 Выдать автомобиль.

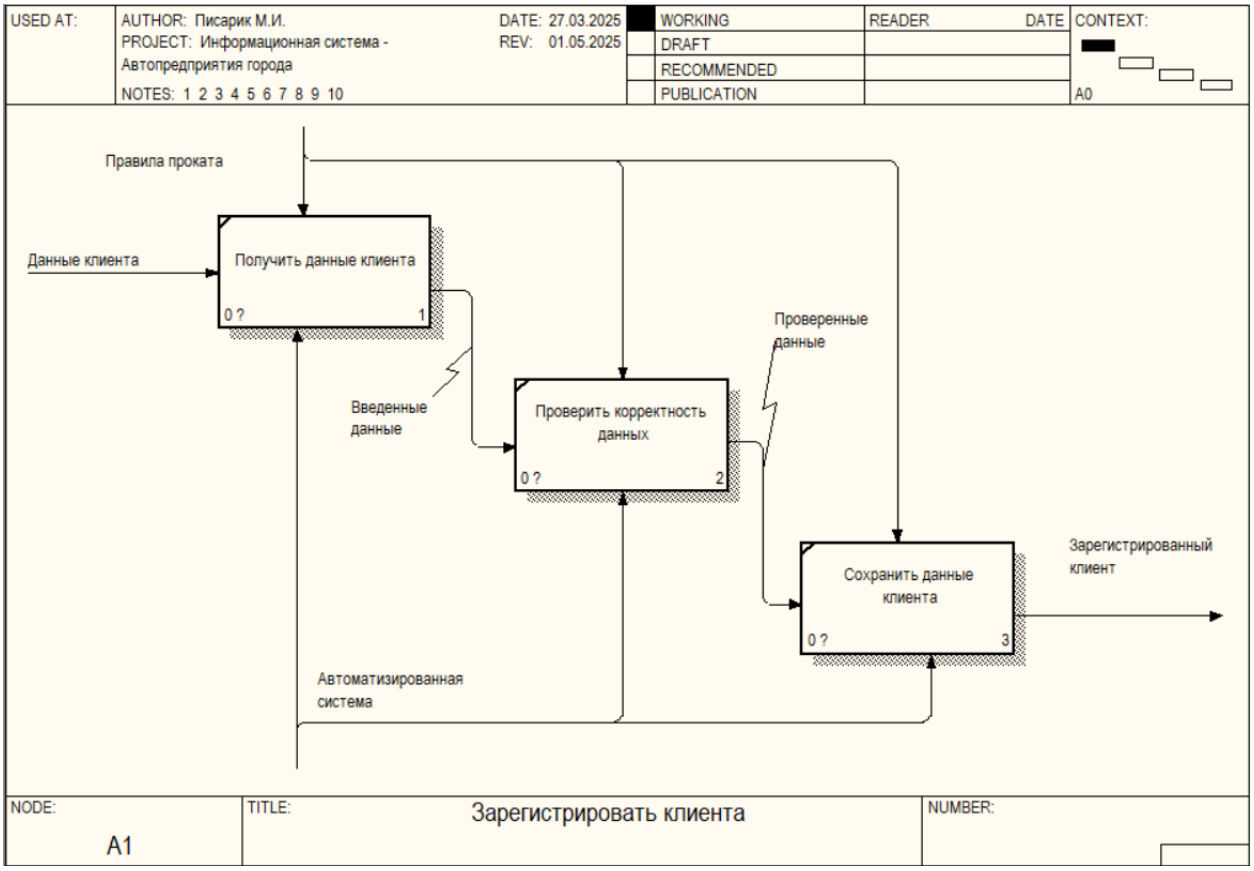

Рисунок 2.3 – блок A1 “Зарегистрировать клиента”

На рисунке 2.3 представлена декомпозиция блока A0 «Предоставить информацию об оценках и посещениях». Каждый из этапов декомпозиции является важным для обеспечения качественного учёта успеваемости студентов и эффективного управления составом университета.

1 Получить данные клиента.
2 Проверить корректность данных.
3 Сохранить данные клиента.

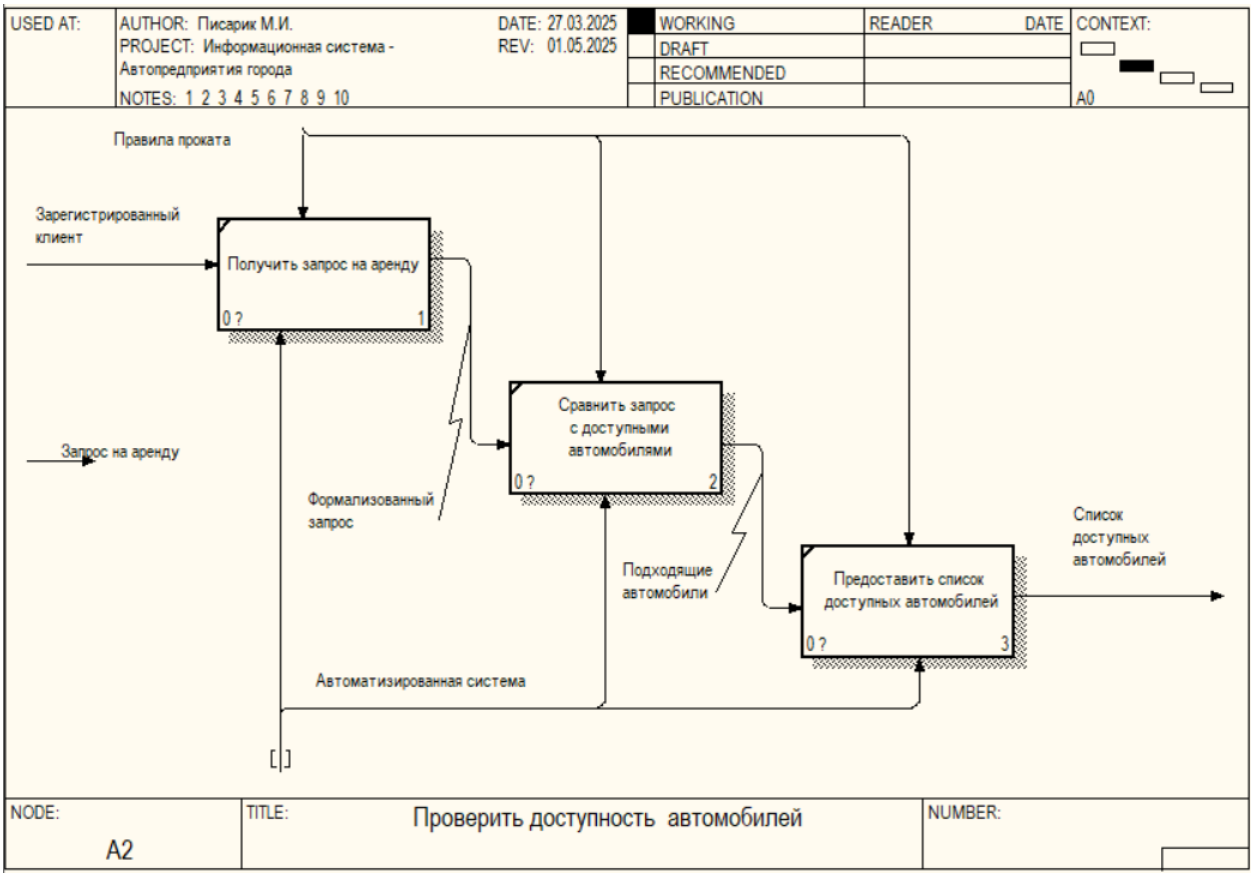

Рисунок 2.4 – блок A2 “Проверить доступность автомобилей ”

На рисунке 2.4 представлена декомпозиция блока A2 «Проверить доступность автомобилей». Каждый из этапов декомпозиции является важным для обеспечения качественного учёта автомобилей и эффективного управления ими.

1 Получить запрос на аренду.
2 Сравнить запрос с доступными автомобилями.
3 Предоставить список доступных автомобилей.

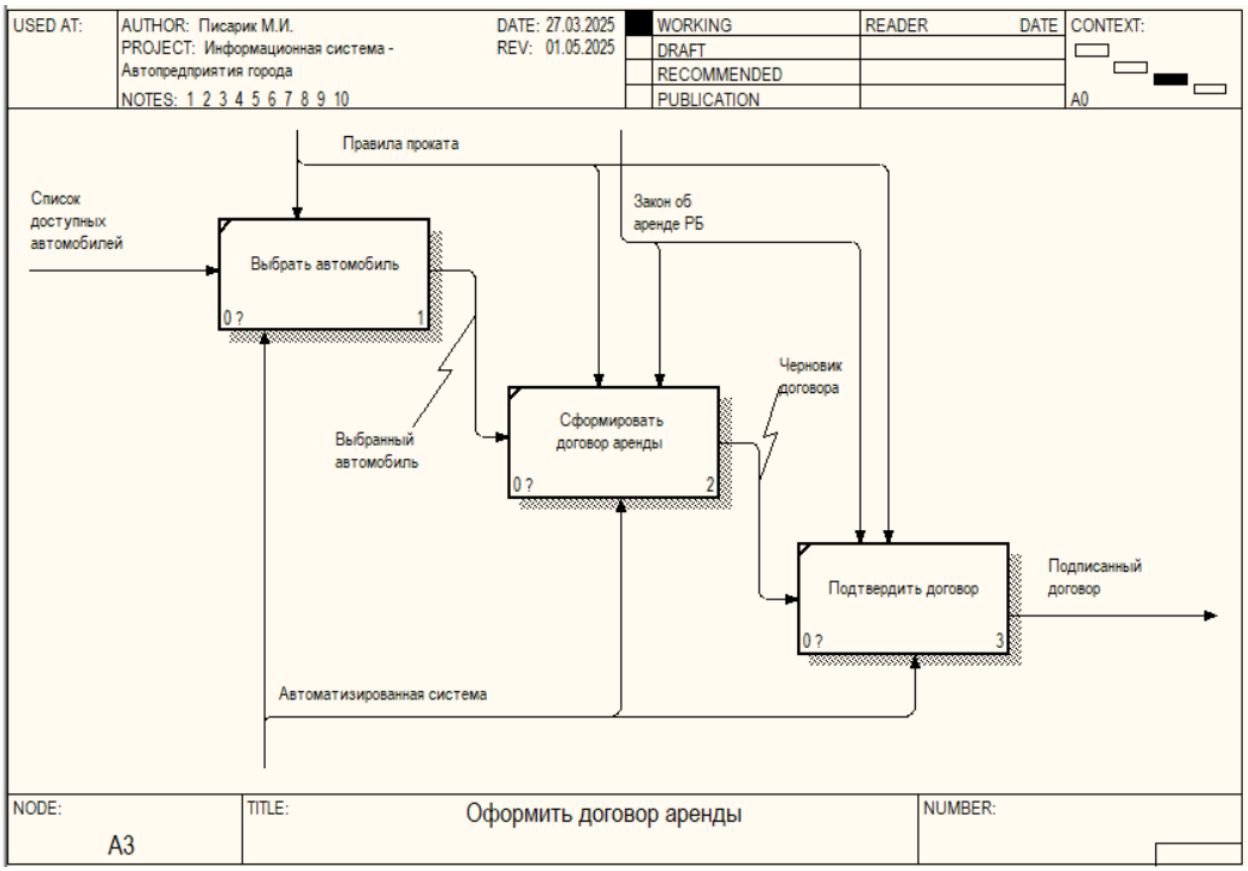

Рисунок 2.5 – блок A3 “Оформить договор аренды ”

На рисунке 2.5 представлена декомпозиция блока A3 «Оформить договор аренды». Каждый из этапов декомпозиции является важным для обеспечения качественного учёта автомобилей и эффективного управления ими.

1 Выбрать автомобиль.
2 Сформировать договор аренды.
3 Подтвердить договор.

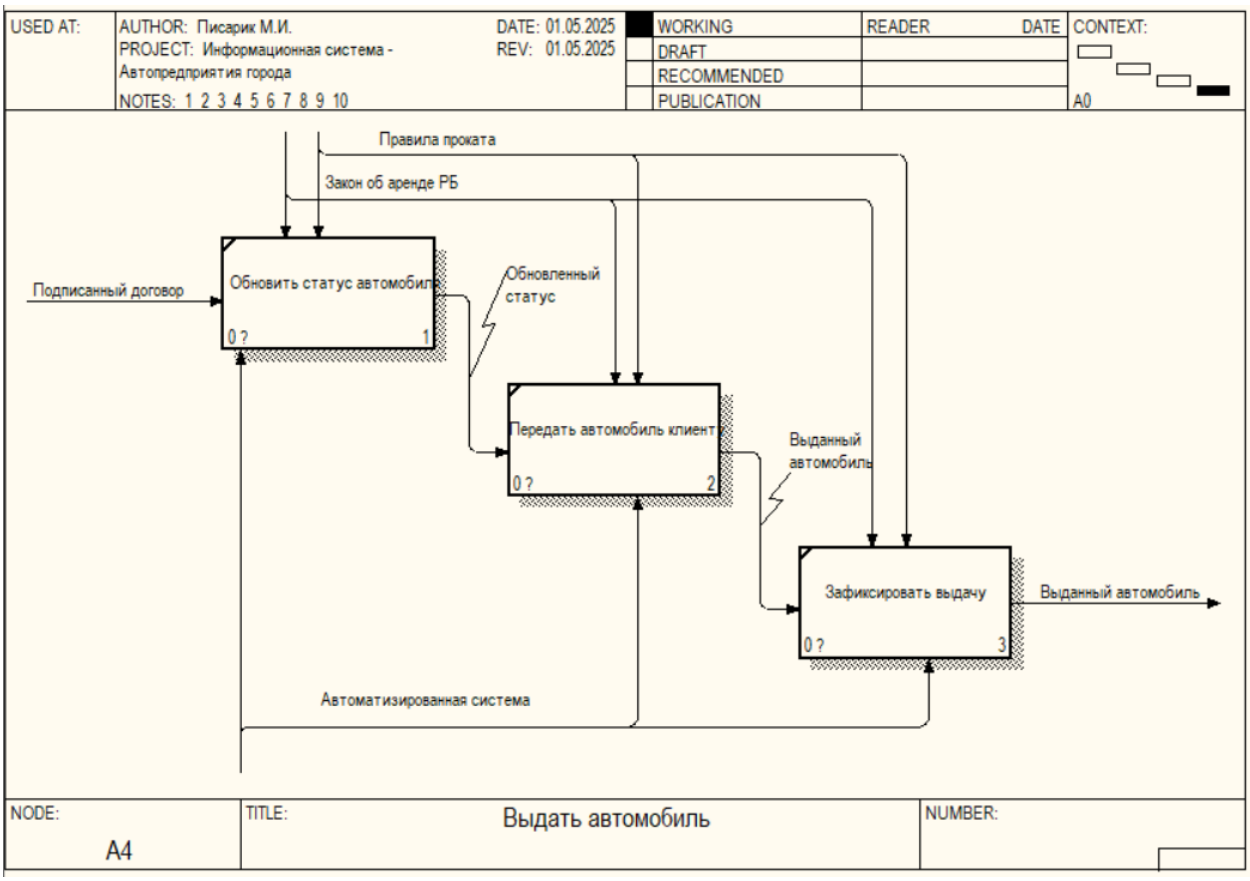

Рисунок 2.6 – блок A4 “Выдать автомобиль ”

На рисунке 6 представлена декомпозиция блока A4 «Отфильтровать данные». Каждый из этапов декомпозиции является важным для обеспечения качественного учёта автомобилей и эффективного управления ими.

1 Обновить статус автомобиля.
2 Передать автомобиль клиенту.
3 Зафиксировать выдачу.

2.3 Описание организации структур, хранящих данные

Для хранения данных в системе используются классы, реализованные на языке C++, которые тесно интегрированы с базой данных SQLite. Таблица

Users предназначен для работы с информацией о пользователях и содержит такие поля, как имя, хешированный пароль для обеспечения безопасности, возраст, опыт вождения, рейтинг, статус блокировки и признак прав администратора. Таблица Cars отвечает за данные об автомобилях и включает поля, такие как тип транспортного средства, модель, базовая цена аренды и минимальный возраст водителя, необходимый для аренды. Все эти данные организованы в соответствующих таблицах SQLite, что обеспечивает быстрый доступ к информации и надежность ее хранения даже при большом объеме запросов.

Таблица 1 – Описание таблицы Car

Название поля	**Тип данных**	**Описание**
ID	INTEGER	Уникальный идентификатор автомобиля (автоинкремент).
type	TEXT	Категория автомобиля (например, Economy, Comfort).
model	TEXT	Модель автомобиля.
basePrice	REAL	Базовая стоимость аренды в день.
minAge	INTEGER	Минимальный возраст для аренды данного автомобиля.

Таблица 2 – Описание таблицы User

Название поля	Тип данных	Описание
ID	INTEGER	Уникальный идентификатор пользователя (автоинкремент).
username	TEXT	Логин пользователя (уникальный).
hashedPassword	TEXT	Хешированный пароль для безопасности.
age	INTEGER	Возраст пользователя в годах.
experience	REAL	Стаж вождения в годах.
rating	INTEGER	Рейтинг пользователя (например, для доверия).
isBlocked	INTEGER	Флаг блокировки пользователя (0 или 1).
isAdmin	INTEGER	Флаг прав администратора (0 или 1).

Эти таблицы обеспечивают структурированное хранение данных, необходимых для функционирования системы проката автомобилей, и позволяют эффективно управлять информацией о пользователях и автопарке.

2.4 Разработка перечня пользовательских функций программы

1. Функция регистрации пользователя

void registerUser()

Обеспечивает создание новой учетной записи в системе. Пользователь вводит имя, пароль, возраст и опыт вождения, после чего данные проверяются на корректность и уникальность имени. Если все условия выполнены, учетная запись сохраняется в базе данных, и пользователю подтверждается успешная регистрация.

2. Функция авторизации пользователя

void loginUser()

Позволяет пользователю войти в систему, вводя имя и пароль. Программа проверяет соответствие введенных данных с хранящимися в базе, и при успешной авторизации предоставляет доступ к соответствующему меню

в зависимости от роли пользователя — обычного пользователя или администратора.

3. Функция отображения доступных автомобилей

void displayAvailableCars(const User& user)

Отображает список автомобилей, доступных для аренды текущему пользователю. Функция учитывает возраст и опыт вождения пользователя, фильтруя транспортные средства по минимальному возрасту и категории, чтобы предложить только подходящие варианты. Информация включает тип, модель и расчетную цену аренды.

4. Функция аренды автомобиля

void rentCar(const User& user)

Обеспечивает процесс аренды выбранного автомобиля. Пользователь выбирает транспортное средство из списка доступных, и система проверяет соответствие его возраста и опыта требованиям автомобиля. После подтверждения аренды информация о сделке фиксируется, и пользователю предоставляется подтверждение.

5. Функция просмотра списка пользователей

void viewUsers()

Доступна администраторам и отображает информацию о всех зарегистрированных пользователях. Функция выводит такие данные, как имя пользователя, возраст, опыт вождения, статус блокировки и наличие прав администратора, что позволяет эффективно контролировать базу пользователей.

6. Функция блокировки/разблокировки пользователя

void blockUnblockUser()

Позволяет администратору блокировать или разблокировать учетную запись пользователя по его имени. Функция проверяет, не является ли пользователь администратором, и изменяет статус блокировки, сохраняя изменения в базе данных с соответствующим уведомлением.

7. Функция просмотра списка автомобилей

void viewCars()

Отображает полный список автомобилей, доступных в системе. Для каждого автомобиля выводятся такие параметры, как тип, модель, базовая цена и минимальный возраст аренды, что помогает администраторам следить за состоянием автопарка.

8. Функция добавления автомобиля

void addCar()

Обеспечивает добавление нового автомобиля в систему администратором. Пользователь выбирает тип автомобиля (Economy, Comfort, Business или Premium), вводит модель и базовую цену, после чего данные

сохраняются в базе данных с автоматическим определением минимального возраста аренды.

9. Функция удаления автомобиля

void removeCar()

Позволяет администратору удалить автомобиль из системы по его модели. После ввода модели система удаляет соответствующую запись из базы данных, подтверждая выполнение операции, если автомобиль был найден.

2.5 Разработка схем алгоритмов работы программы

//////////////////////

3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного средства является процессом, направленным на проверку соответствия фактического программного продукта заданным требованиям. Оно также необходимо для обнаружения и исправления дефектов в продукте.

Функциональные спецификации программного средства приведены в разделе 1.4 и описываются вариантами использования.

Результаты тестирования показали, что программное средство для проката автомобилей соответствует функциональным требованиям, и все функции работают корректно.

В программе реализованы проверки на правильность ввода целочисленных значений и недопустимых символов при заполнении полей. Эти проверки гарантируют, что недопустимые значения не будут приняты. В случае некорректного ввода программа очищает поле ввода и запрашивает повторный ввод. Этот процесс повторяется до тех пор, пока пользователь не введет допустимое значение.

Во время авторизации при попытке ввода неправильного логина и/или пароля выводится сообщение о том, что вход не может быть выполнен (рисунок 3.1).

Рисунок 3.1 – Сообщение об ошибке входа

При попытке регистрации пользователя уже с существующим именем – выдает следующую ошибку (рисунок 3.2)

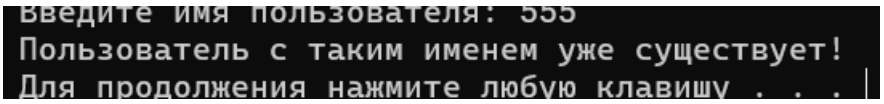

Рисунок 3.2 – Существующее имя пользователя

При регистрации и авторизации могут возникнуть следующие исключительные ситуации: пароль состоит меньше чем из 6 символов, логин содержит пробелы, возраст клиента меньше чем 18 лет, стаж управления автомобилем некорректен (рисунки 3.4-3.6).

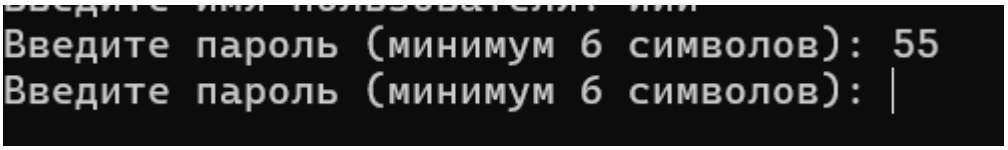

Рисунок 3.4 – Проверка пароля

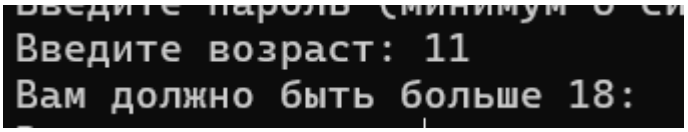

Рисунок 3.5 – Проверка возраста

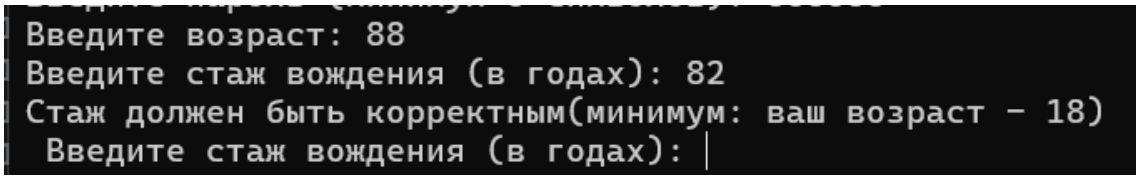

Рисунок 3.6 – Проверка стажа

Проведение тестирования программного обеспечения подчеркивает его важность и преимущества. Тестирование помогает выявить потенциальные ошибки и исключительные ситуации, которые могут возникнуть при использовании программы. Это способствует обеспечению надежности и безопасности программного средства, а также повышает удовлетворенность пользователей.

4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ТЕСТОВЫЙ ПРИМЕР

4.1 Авторизация

Приложение запускается файлом courseWorkVer0.09.exe. После запуска открывается главное окно со меню первого уровня (рисунок 4.1).

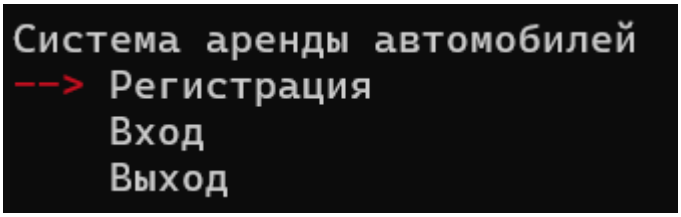

Рисунок 4.1– Меню первого уровня

Если же вход пройдет удачно, администратору будет предложен ряд функций, который описан в меню второго уровня (рисунок 4.4). Учетная запись «admin» принадлежит администратору. Модуль администратора рассмотрен в следующей главе.

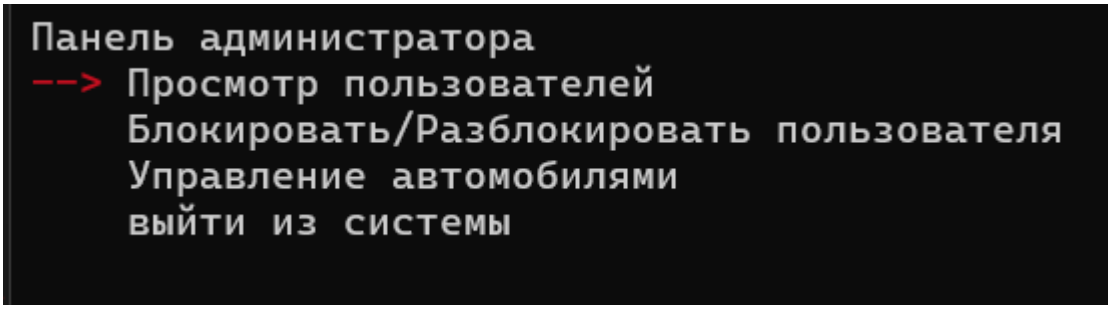

Рисунок 4.4 – Меню второго уровня для администратора

Авторизация пользователя происходит аналогично, единственное различие – меню второго уровня для пользователя (рисунок 4.5).

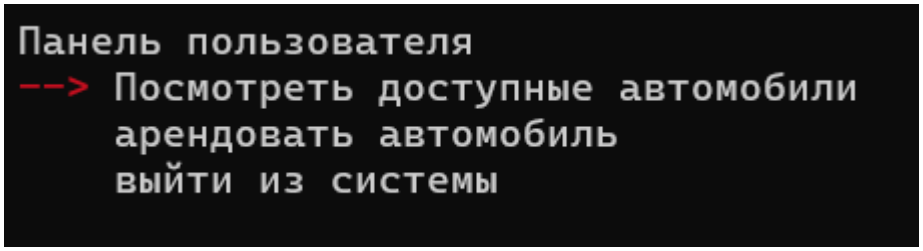

Рисунок 4.5 – Меню второго уровня для пользователя

4.2 Модуль администратора.

При входе в систему в роли администратора перед пользователем появится главное меню администратора второго уровня (рисунок 4.4), которое включает такие пункты, как:

- Просмотреть пользователей ;
- Блокировать или разблокировать пользователя;
- Управление автомобилями;
- Выйти из системы.

При просмотре всех учетных записей, программа выводит данные в табличном виде (рисунок 4.6).

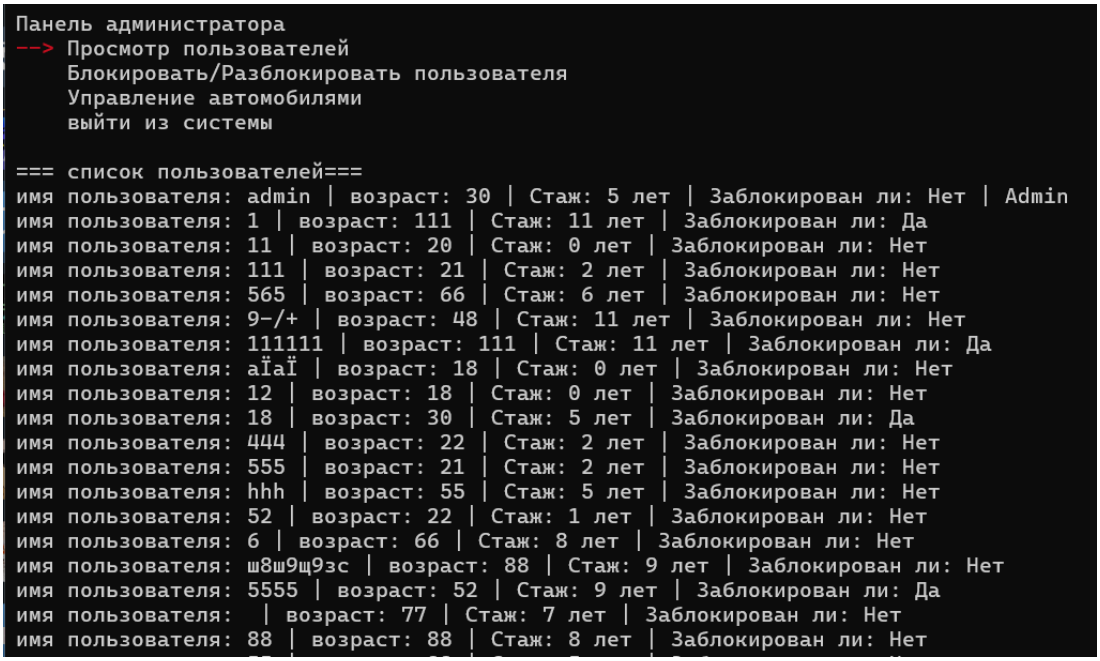

Рисунок 4.6 – Просмотр всех учетных записей

При необходимости заблокировать какую-либо учетную запись, программа запрашивает логин пользователя, а затем блокирует его (рисунок 4.7).

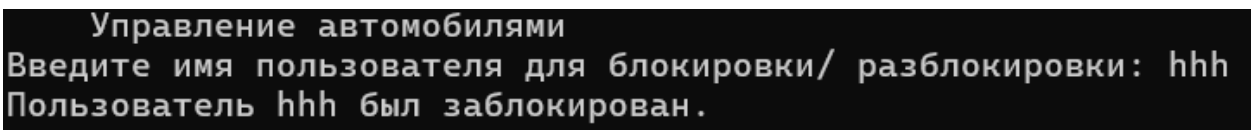

Рисунок 4.7 – Блокировка пользователя

При выборе функции «Выбрать автомобиль для добавления» администратору предоставляется возможность добавлять автомобили (рисунок 4.8-4.9).

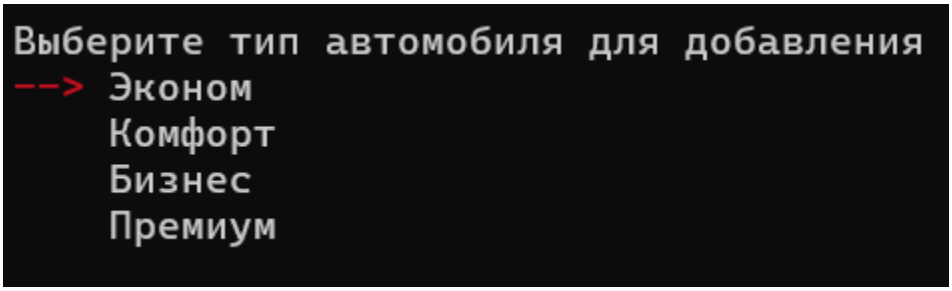

Рисунок 4.8 – Выбрать тип автомобиля для добавления

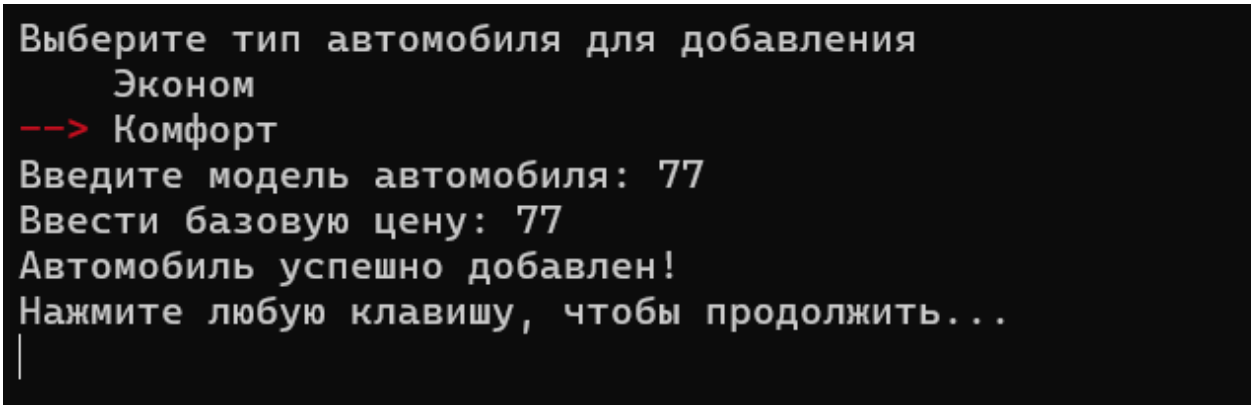

Рисунок 4.9 –Ввести параметры автомобиля для добавления

При выборе функции «Выбрать автомобиль для удаления» администратору предоставляется возможность удалять автомобили (рисунок 4.10).

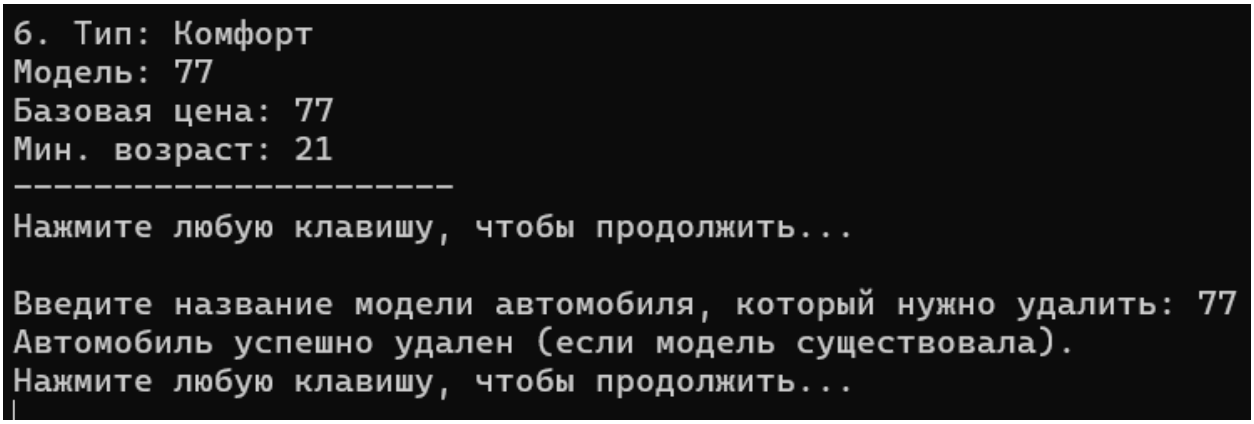

Рисунок 4.10 – Выбрать автомобиль для удаления

4.3 Модуль пользователя

При запуске программы пользователь может как авторизоваться, так и зарегистрироваться, если пользователь использует программное средство впервые.

При успешной авторизации появляется меню второго уровня для пользователя (рисунок 4.5)

При выборе функции «Просмотреть доступные автомобили» пользователю предоставляется весь доступный список автомобили и требований к ним (рисунок 4.11).

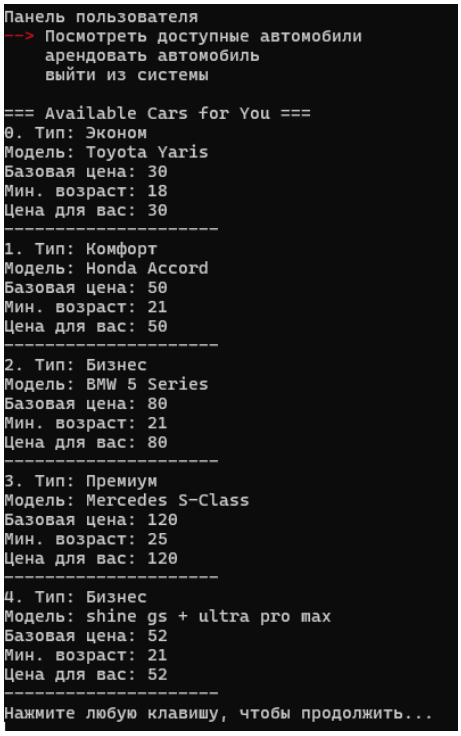

Рисунок 4.11 – Просмотреть доступные автомобили

При выборе функции «Выбрать автомобиль для аренды» пользователю предоставляется весь доступный список автомобили и возможность арендовать их (рисунок 4.11).

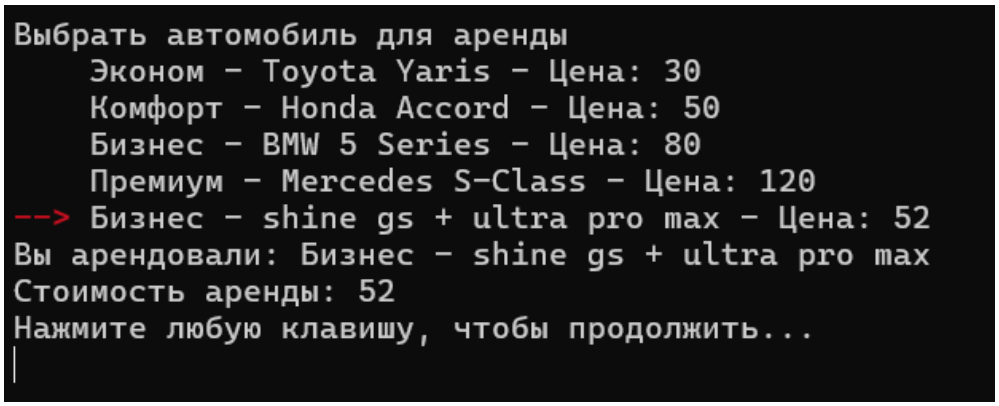

Рисунок 4.12 – Выбрать автомобиль для аренды

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была создана программа для автоматизированного проката автомобилей в программной среде Microsoft Visual Studio 2022. В целях автоматизации и системности учета было разработано программное средство, которое позволяет заметно упростить деятельность организаций занимающихся сдачей авто а в аренду.

Разработанное программное средство обеспечивает выполнение следующих функций:

– ввод и хранение исходных данных для формирования промежуточных и результатных данных:база данных пользователей и автомашин;

– редактирование данных: добавление и удаление автомобилей в системе, внесение изменений в статус пользователя.

Система имеет приятный и доступный интерфейс, полный набор функций, необходимых для удобной аренды авто.

Для достижения поставленной цели успешно решены следующие задачи:

– исследованы основные моменты функционирования компаний-арендодателей;

– проанализирована деятельность управляющего организацией, которая занимается сдачей автомобилей напрокат;

Выполнено проектирование и разработка программного средства: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд IDEF0-диаграмм; описаны основные алгоритмы работы программы; разработано руководство пользователя; выполнено тестирование системы, показавшее ее соответствие функциональным требованиям, поставленным в задании на разработку.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.

ПРИЛОЖЕНИЕ А

Листинг кода алгоритмов, реализующих бизнес-логику

```
#include "App.h"
#include "DataStorage.h"
#include "Utilities.h"
#include "InputChecker.h"
#include "termcolor.hpp"
#include <iostream>
#include <conio.h>
#include <windows.h>
#include <sstream>

using namespace std;

int showMenu(const std::string& title, const std::string options[],
    int optionCount) {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_CURSOR_INFO originalCursorInfo;
    GetConsoleCursorInfo(hConsole, &originalCursorInfo);

    CONSOLE_CURSOR_INFO cursorInfo = originalCursorInfo;
    cursorInfo.bVisible = false;
    SetConsoleCursorInfo(hConsole, &cursorInfo);
    system("cls");
    std::cout << title << std::endl;
    while (_kbhit()) _getch();
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(hConsole, &csbi);
    SHORT menuStartY = csbi.dwCursorPosition.Y;
    for (int i = 0; i < optionCount; ++i) {
        if (i == 0) {
            std::cout << termcolor::red << "--> " << termcolor::reset;
        }
        else {
            std::cout << "    ";
        }
        std::cout << options[i] << std::endl;
```

Продолжение приложения А

```
  }
   int choice = 0;
   int previousChoice = 0;
   GetConsoleScreenBufferInfo(hConsole, &csbi);
   int consoleWidth = csbi.dwSize.X;
   //  TODO paint console output
   while (true) {
      if (choice != previousChoice) {

         SetConsoleCursorPosition(
            hConsole, { 0, static_cast<SHORT>(menuStartY + previousChoice) });
         std::cout << std::string(consoleWidth, ' ') << termcolor::grey << "\r    "
            << termcolor::reset << options[previousChoice];

         SetConsoleCursorPosition(
            hConsole, { 0, static_cast<SHORT>(menuStartY + choice) });
         std::cout << std::string(consoleWidth, ' ') << termcolor::red << "\r--> "
            << termcolor::reset << options[choice];

         previousChoice = choice;
      }

      int key = _getch();
      if (key == 224) {
         key = _getch();
         if (key == 72 && choice > 0)
            choice--;
         else if (key == 80 && choice < optionCount - 1)
            choice++;
      }
      else if (key == 13) { // Enter
         SetConsoleCursorInfo(hConsole, &originalCursorInfo);
         while (_kbhit()) _getch();
         return choice;
      }
   }
}


App::App() {
   auto& db = DataStorage::getInstance();
```

Продолжение приложения А

```
auto cars = db.getCars();
   if (cars.empty()) {
      db.addCar(make_shared<EconomyCar>("Toyota Yaris", 30));
```

```
        db.addCar(make_shared<ComfortCar>("Honda Accord", 50));
        db.addCar(make_shared<BusinessCar>("BMW 5 Series", 80));
        db.addCar(make_shared<PremiumCar>("Mercedes S-Class", 120));
    }

    if (db.findUserByName("admin") == nullptr) {
        db.addUser(User("admin", "admin", 30, 10, 0, false, true));
    }
}


void App::run() {
    bool exitFlag = false;
    while (!exitFlag) {
        const string menu[] = { "Регистрация", "Вход", "Выход" };
        int choice = showMenu("Система аренды автомобилей", menu, 3);
        switch (choice) {
        case 0: registerUser(); break;
        case 1: loginUser(); break;
        case 2: exitFlag = true; break;
        }
    }
}


void App::registerUser() {
    cin.ignore();
    string username, password, inputAge, inputExp;
    int age = 0;
    double exp = -1;

    cout << "\n=== Регистрация ===\n";
    cout << "Введите имя пользователя: ";
    getline(cin, username);

    if (DataStorage::getInstance().findUserByName(username)) {
        cout << "Пользователь с таким именем уже существует!\n";
        system("pause");
        return;
    }
```



```
do {
        cout << "Введите пароль (минимум 6 символов): ";
        getline(cin, password);
```

```
    } while (!isValidPassword(password));
do {
        cout << "Введите возраст: ";
        getline(cin, inputAge);
        if (inputChecker::isIntCorrect(inputAge)) {
            age = stoi(inputAge);
            if (age < 18  age > 100) {
                age = 0;
                cout << "Вам должно быть больше 18:\n";
            }

        }

    } while (age == 0);

    do {
        cout << "Введите стаж вождения (в годах): ";
        getline(cin, inputExp);
        if (inputChecker::isDoubleCorrect(inputExp)) {
            exp = stod(inputExp);
            if (exp < 0  exp >(age - 18)) exp = -1;
            {
                cout << "Стаж должен быть корректным(минимум: ваш возраст - 18)\n ";
            }
        }
    } while (exp == -1);

    User user(username, password, age, exp);
    if (DataStorage::getInstance().addUser(user)) {
        cout << "Пользователь успешно зарегистрирован!" << endl;
    }
    else {
        cout << "Ошибка регистрации." << endl;
    }

    system("pause");
}


void App::loginUser() {
    cin.ignore();
```



```
 string username, password;

    cout << "\n=== вход ===\n";
```

```
    cout << "введите имя пользователя: ";
    getline(cin, username);
    cout << "введите пароль: ";
    getline(cin, password);

    User* user = DataStorage::getInstance().findUserByName(username);
    if (user == nullptr || !user->checkPassword(password)) {
        cout << "Неверное имя пользователя или пароль." << endl;
        system("pause");
        if (user)
            delete user;
        return;
    }

    if (user->isBlocked) {
        cout << "Ваш аккаунт заблокирован." << endl;
        system("pause");
        delete user;
        return;
    }

    if (user->isAdmin) {
        adminMenu(user);
    }
    else {
        userMenu(user);
    }

    delete user;
}
void App::userMenu(User* user) {
    bool exitUser = false;
    while (!exitUser) {
        const string userOptions[] = { "Посмотреть доступные автомобили", "арендовать
автомобиль",
                                    "выйти из системы" };
        int choice = showMenu("Панель пользователя", userOptions, 3);
        switch (choice) {
        case 0:
            displayAvailableCars(*user);
            break;
```

Продолжение приложения А

```
  case 1:
            rentCar(*user);
            break;
```

```
        case 2:
            exitUser = true;
            break;
        }
    }
}
void App::adminMenu(User* admin) {
    bool exitAdmin = false;
    while (!exitAdmin) {
        const string adminOptions[] = { "Просмотр пользователей",
"Блокировать/Разблокировать пользователя",
                                "Управление автомобилями", "выйти из системы" };
        int choice = showMenu("Панель администратора", adminOptions, 4);
        switch (choice) {
        case 0:
            viewUsers();
            break;
        case 1:
            blockUnblockUser();
            break;
        case 2:
            manageCars();
            break;
        case 3:
            exitAdmin = true;
            break;

        }
    }
}
void App::addCar() {
    cin.ignore();
    const string carTypes[] = { "Эконом", "Комфорт", "Бизнес", "Премиум" };
    int typeChoice = showMenu("Выберите тип автомобиля для добавления", carTypes, 4);
    string model;
    double price;
    cout << "\nВведите модель автомобиля: ";
    getline(cin, model);
    cout << "Ввести базовую цену: ";
    cin >> price;
    cin.ignore();
```



```
shared_ptr<Car> newCar;
    switch (typeChoice) {
    case 0:
```

```
            newCar = make_shared<EconomyCar>(model, price);
            break;
        case 1:
            newCar = make_shared<ComfortCar>(model, price);
            break;
        case 2:
            newCar = make_shared<BusinessCar>(model, price);
            break;
        case 3:
            newCar = make_shared<PremiumCar>(model, price);
            break;
        }
        DataStorage::getInstance().addCar(newCar);
        cout << "Автомобиль успешно добавлен!" << endl;
        cout << "Нажмите любую клавишу, чтобы продолжить..." << endl;
        cin.get();
    }
    void App::viewCars() {
        cout << "\n=== доступные автомобили ===\n";
        vector<shared_ptr<Car>> cars = DataStorage::getInstance().getCars();
        int index = 0;
        for (auto& car : cars) {
            cout << index++ << ". ";
            car->displayInfo();
            cout << dashesManip << "\n";
        }
        cout << "Нажмите любую клавишу, чтобы продолжить..." << endl;
        cin.get();
    }
    void App::removeCar() {
        viewCars();
        cout << "Введите название модели автомобиля, который нужно удалить: ";
        string model;
        getline(cin, model);
        DataStorage::getInstance().removeCar(model);
        cout << "Автомобиль успешно удален (если модель существовала)." << endl;
        cout << "Нажмите любую клавишу, чтобы продолжить..." << endl;
        cin.get();
    }
    void App::displayAvailableCars(const User& user) {
        cout << "\n=== Available Cars for You ===\n";
```



```
     vector<shared_ptr<Car>> cars = DataStorage::getInstance().getCars();
        vector<shared_ptr<Car>> availableCars;
        for (auto& car : cars) {
```

```
        if (user.age < car->getMinAge())
            continue;
        if (user.experience < 1.0 && car->getType() != "Economy")
            continue;
        availableCars.push_back(car);
    }
    if (availableCars.empty()) {
        cout << "Нет машин, которые бы отвечали критериям возраста/опыта."
            << endl;
    }
    else {
        int idx = 0;
        for (auto& car : availableCars) {
            cout << idx++ << ". ";
            car->displayInfo();
            cout << "Цена для вас: " << car->getPrice(user) << "\n";
            cout << "--------------------\n";
        }
    }
    cout << "Нажмите любую клавишу, чтобы продолжить..." << endl;
    cin.get();
}
void App::rentCar(const User& user) {
    cout << "\n=== арендовать машину ===\n";
    vector<shared_ptr<Car>> cars = DataStorage::getInstance().getCars();
    vector<shared_ptr<Car>> availableCars;
    for (auto& car : cars) {
        if (user.age < car->getMinAge())
            continue;
        if (user.experience < 1.0 && car->getType() != "Эконом")
            continue;
        availableCars.push_back(car);
    }
    if (availableCars.empty()) {
        cout << "Нет машин, которые бы отвечали критериям возраста/опыта."
            << endl;
        cout << "Нажмите любую клавишу, чтобы продолжить..." << endl;
        cin.get();
        return;
    }
    vector<string> carOptions;
```

Продолжение приложения А

```
for (auto& car : availableCars) {
        stringstream ss;
        ss << car->getType() << " - " << car->getModel()
```

```
            << " - Цена: " << car->getPrice(user);
        carOptions.push_back(ss.str());
    }
    int choice = showMenu("Выбрать автомобиль для аренды", carOptions.data(),
        carOptions.size());
    cout << "\nВы арендовали: " << availableCars[choice]->getType() << " - "
        << availableCars[choice]->getModel() << "\n";
    cout << "Стоимость аренды: " << availableCars[choice]->getPrice(user)
        << "\n";
    cout << "Нажмите любую клавишу, чтобы продолжить..." << endl;
    cin.get();
}
void App::viewUsers() {
    cout << "\n=== список пользователей===\n";
    vector<User> users = DataStorage::getInstance().getUsers();
    for (auto& u : users) {
        cout << "имя пользователя: " << u.username << " | возраст: " << u.age
            << " | Стаж: " << u.experience << " лет"
            << " | Заблокирован ли: " << (u.isBlocked ? "Да" : "Нет")
            << (u.isAdmin ? " | Admin" : "") << endl;
    }
    cout << "Нажмите любую клавишу, чтобы продолжить..." << endl;
    cin.get();
}
void App::blockUnblockUser() {
    cin.ignore();
    cout << "\nВведите имя пользователя для блокировки/ разблокировки: ";
    string uname;
    getline(cin, uname);
    User* user = DataStorage::getInstance().findUserByName(uname);
    if (user == nullptr) {
        cout << "Пользователь не найден." << endl;
        return;
    }
    if (user->isAdmin) {
        cout << "Невозможно изменить учетную запись администратора." << endl;
        delete user;
        return;
    }
    user->isBlocked = !user->isBlocked;
    DataStorage::getInstance().updateUser(*user);
```

Продолжение приложения А

```
cout << "Пользователь " << user->username
        << (user->isBlocked ? " был заблокирован."
            : " был разблокирован.")
```

```
        << endl;
    cout << "Нажмите любую клавишу, чтобы продолжить..." << endl;
    cin.get();
    delete user;
}
void App::manageCars() {
    bool exitCars = false;
    while (!exitCars) {
        const string carOptions[] = { "Просмотр автомобилей", "Добавить автомобиль",
"Удалить автомобиль",
                                      "назад" };
        int choice = showMenu("Панель изменения данных об авто", carOptions, 4);
        switch (choice) {
        case 0:
            viewCars();
            break;
        case 1:
            addCar();
            break;
        case 2:
            removeCar();
            break;
        case 3:
            exitCars = true;
            break;
        }
    }
}
```

ПРИЛОЖЕНИЕ Б

кикик

ПРИЛОЖЕНИЕ В

кикик