МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра математического и компьютерного моделирования

# РАЗРАБОТКА ПРОТОТИПА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УЧЕТА ВЗАИМОДЕЙСТВИЯ С КЛИЕНТАМИ ДЛЯ ПРЕДПРИЯТИЯ СФЕРЫ УСЛУГ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ


студента 5 курса 561 группы

направление 09.03.03 — Прикладная информатика

механико-математического факультета

Ефремова Дмитрия Леонидовича


Научный руководитель
доцент, к.ф.-м.н. _____ О.С. Кузнецова
подпись, дата

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент _____ Ю.А. Блинков
подпись, дата

Саратов 2025

# ВВЕДЕНИЕ

**Целью бакалаврской работы** является разработка прототипа автоматизированной системы учета взаимодействия с клиентами для предприятий сферы услуг, основанной на принципах CRM (Customer Relationship Management), с использованием современных технологий и подходов к управлению взаимоотношениями с клиентами.

**Объект исследования** – автоматизированные системы учета взаимодействия с клиентами (CRM), которые являются программными решениями, используемые компаниями для управления и оптимизации взаимодействия с клиентами.

**Предмет исследования** – анализ основных принципов автоматизированных систем учета взаимодействия с клиентами (CRM-систем).

Для достижения поставленных целей в работе необходимо решить следующие **задачи**:

- определить основные понятия, необходимые для описания автоматизированных систем учета взаимодействия с клиентами;
- проанализировать существующие типы автоматизированных систем учета взаимодействия с клиентами;
- Разработать техническое задание для CRM-системы;
- Спроектировать архитектуру приложения;
- Реализовать прототип CRM-системы;

**Актуальность темы.** Актуальность темы выпускной квалификационной работы «Разработка прототипа автоматизированной системы учета взаимодействия с клиентами » обусловлена несколькими ключевыми факторами. Во-первых, в условиях растущего конкуренции и высокого уровня ожиданий со стороны клиентов, компании все больше нуждаются в инструментах, которые позволят им оптимизировать процессы взаимодействия с клиентами, улучшить качество обслуживания и повысить уровень удовлетворенности клиентов. Во-вторых, с развитием технологий появляются новые возможности для автоматизации и оптимизации бизнес-процессов, включая управление взаимоотношениями с клиентами. CRM-системы, как одна из таких технологий, становятся все более востребованными и важными для успешного веде-

ния бизнеса. В-третьих, в эпоху цифровых технологий и большого количества информации о клиентах, возможность персонализировать взаимодействие с клиентами становится критически важной.

Также актуальность темы заключается в возможности, с помощью CRM-системы, компаниям собирать, анализировать и использовать данные о клиентах для создания персонализированных предложений и улучшения взаимодействия.

В целом, данная работа актуальна и важна для развития бизнеса и технологий, а также для улучшения качества взаимодействия с клиентами.

**Практическая значимость** проводимого исследования состоит в том, что полученные результаты и разработанный прототип позволяют предприятиям сферы услуг значительно повысить эффективность работы с клиентами, оптимизировать процессы управления взаимоотношениями и повысить уровень удовлетворенности клиентов благодаря централизации данных, автоматизации процессов и аналитике.

**Структура и содержание бакалаврской работы.**Работа состоит из введения, 3 разделов, заключения, списка использованных источников, содержащего 20 наименований и приложения. Общий объем работы составляет 57 страниц.

## ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обосновывается актуальность темы работы, формулируется цель работы и решаемые задачи, отмечается практическая значимость полученных результатов.

В **первом** разделе рассматриваются основные принципы, типы, Функциональность, недостатки и ограничения CRM-систем.

CRM (Customer Relationship Management) — это система управления взаимоотношениями с клиентами, которая включает в себя набор инструментов и методик для управления всеми взаимодействиями с клиентами на протяжении всего жизненного цикла продукта или услуги.

Основные принципы CRM включают:

1. Централизация данных.

2. Интеграция процессов.
3. Предоставление персонализированного обслуживания.
4. Анализ и прогнозирование.
5. Улучшение взаимодействия с клиентами.

**Типы CRM-систем**

**Аналитические CRM-системы** помогают компаниям лучше использовать информацию, которую они собирают о клиентах. Это может включать предпочтения клиентов, каналы связи, точки контакта, интересы и многое другое. В то время как операционные CRM-системы, например, приписывают клиентов к конвейеру продаж, аналитические CRM-системы помогают понять их. Затем, на основе данных контактов, компоненты для добычи данных могут собрать еще больше информации, включая тренды, и помочь распознать шаблоны в наборах данных. Таким образом, компании могут использовать данные, которые уже собирают, для принятия более обоснованных бизнес-решений.

**Коллективные CRM-системы** направлены на координацию и синхронизацию работы различных отделов компании, включая продаж, маркетинг, обслуживание клиентов и другие. Эти системы обеспечивают единый источник данных и инструменты для совместной работы над проектами и задачами, что способствует повышению эффективности и согласованности действий внутри организации.

**Облачные CRM-системы** предоставляют доступ к функционалу CRM через интернет, что позволяет пользователям работать с системой из любого места и на любом устройстве. Эти системы обычно предлагают гибкие планы подписки, что делает их доступными для малых и средних предприятий. Облачные CRM обеспечивают высокую доступность, безопасность данных и легкость обновления.

**Коробочные CRM-системы** представляют собой комплексные решения, включающие в себя не только программное обеспечение для управления взаимоотношениями с клиентами, но и дополнительные услуги, такие как консультации, интеграция с другими системами и поддержка. Эти решения предлагаются как готовые к использованию, что упрощает процесс внедрения и адаптации системы к специфике бизнеса компании.

Разработанный прототип будет основываться на принципах коробочных CRM-систем. Такой подход упрощает процесс внедрения и адаптации системы, сокращая время до начала эксплуатации и снижая риски связанные с ошибками в процессе разработки и внедрения.

**Функциональность CRM-систем**

**Управление контактами** включает в себя сбор, хранение и анализ информации о клиентах и потенциальных клиентах. Это позволяет компаниям иметь полное представление о своих клиентах, их предпочтениях, истории взаимодействия и других важных деталях, что способствует улучшению взаимоотношений и повышению уровня обслуживания.

**Управление сделками** включает в себя отслеживание всех этапов процесса продажи, от первого контакта с потенциальным клиентом до закрытия сделки. Это помогает компаниям оптимизировать процессы продаж, улучшить эффективность работы с клиентами и повысить уровень удовлетворенности клиентов.

**Маркетинговые функции CRM-систем** позволяют компаниям анализировать данные о клиентах и целевых группах для разработки и реализации эффективных маркетинговых кампаний. Это включает в себя создание персонализированных предложений, проведение рекламных кампаний и анализ результатов маркетинговых активностей.

**Функции обслуживания клиентов** включают в себя автоматизацию взаимодействия с клиентами, от ответов на вопросы до решения жалоб и проблем. Это помогает улучшить качество обслуживания, сократить время ответа на запросы и повысить уровень удовлетворенности клиентов.

**Интеграция с другими системами**, такими как ERP, бухгалтерия, системы управления контентом и другие, позволяет компаниям обеспечить бесперебойное взаимодействие между различными отделами и процессами. Это способствует повышению эффективности работы и улучшению координации действий.

**Мобильные функции CRM** позволяют сотрудникам компании работать с системой из любого места и на любом устройстве. Это обеспечивает гибкость и удобство использования, а также позволяет сотрудникам быть всегда в курсе последних обновлений и изменений.

**Безопасность и конфиденциальность** являются ключевыми аспектами функциональности CRM-систем. Компании должны обеспечивать защиту данных клиентов от несанкционированного доступа, а также соблюдать законодательство о защите данных. Это включает в себя использование шифрования, двухфакторной аутентификации и других мер безопасности.

**Недостатки CRM-систем** включают высокую стоимость внедрения, сопротивление сотрудников и необходимость специального обучения, что может замедлить процесс внедрения и снижать эффективность использования системы.

**Второй** раздел посвящен проектированию приложения.

**Техническое задание**

Программа должна представлять собой приложение для управления системой продаж. Оно должно быть разработано для использования администратором и клиентом в компании. Главная цель программы - обеспечить эффективное управление товарами, клиентами и заказами.

Программа должна обладать следующими функциями:

1. Вход для администратора и клиента.
2. Добавление товаров.
3. Регистрация клиентов-покупателей.
4. Добавление клиентом товаров в корзину.
5. Оформление заказа.
6. Лист заказов у администратора.
7. Хранение данных в базе данных SQL.
8. Раздел статистики.

**Архитектура приложения**

Для реализации приложения был использован архитектурный шаблон проектирования MVC.

Модель-представление-контроллер (MVC) — это архитектурный шаблон проектирования, который разделяет приложение на три основные компоненты: модель, представление и контроллер. Этот шаблон широко используется для создания масштабируемых и расширяемых проектов.

— **Модель (Model).** Компонент модели представляет данные и бизнес-логику приложения. Он отвечает за управление данными приложения,

обработку бизнес-правил и реагирование на запросы информации от других компонентов, таких как представление и контроллер.

— **Представление (View).** Представление отвечает за интерфейс пользователя, который представляет информацию пользователю и принимает от него. Оно взаимодействует с контроллером для получения данных из модели и обновления своего отображения.

— **Контроллер (Controller).** Контроллер действует как посредник между моделью и представлением. Он обрабатывает ввод пользователя, обновляет модель соответствующим образом и обновляет представление для отражения изменений в модели. В контроллере содержится логика приложения, такая как валидация входных данных и преобразование данных.

**Моделирование архитектуры**

**Выделение Сущностей**

Первым шагом является определение основных сущностей, которые будут использоваться в приложении:

1. `User` представляет собой модель данных для пользователя в системе. Содержит: уникальный идентификатор пользователя, имя пользователя, пароль пользователя.
2. `Order` представляет собой модель данных для заказа в системе. Содержит: артикул товара, идентификатор пользователя, дату и время создания заказа.
3. `Buyer` представляет собой модель данных для покупателя в системе. Содержит: имя покупателя, фамилию, отчество, номер телефона, дату рождения, дату регистрации, идентификатор покупателя.
4. `Product` представляет собой модель данных для товара в системе. Содержит: название, `URL` изображения товара, цену, артикул.

Вторым шагом является определение функциональности контроллеров (Controller), которые и будут производить всю бизнес-логику в данном приложении.

**UserController** должен обладать следующим функционалом:

1. инициализация и загрузка пользователей;
2. вход в систему;

3. добавление пользователя;
4. удаление пользователя;

**ProductController** должен обладать следующим функционалом:

1. инициализация и загрузка товаров;
2. добавление товаров;
3. общая структура для разделения временного (корзина) и постоянного хранения товаров.;

**OrderController** должен обладать следующим функционалом:

1. инициализация и загрузка заказов;
2. добавление заказа;
3. общая структура хранения данных;

**BuyerController** должен обладать следующим функционалом:

1. инициализация и загрузка покупателей;
2. добавление покупателей;
3. общая структура хранения данных;

**Представление (View)**

Третим шагом является определение визуализации представлений (View). Необходимые View в приложении:

1. Экран входа и регистрации. Этот экран является точкой входа в систему для пользователей. Он должен включать следующие компоненты:
   - Форму входа: Поля для ввода логина и пароля пользователя, а также кнопку "Войти".
   - Форму регистрации: Поле для ввода нового логина и пароля и дополнительных данных, а также кнопку "Зарегистрироваться".
   - Информационные сообщения: Отображение успешных или ошибочных сообщений после выполнения операций входа или регистрации.
2. Страница администратора. Эта страница предназначена для управления различными аспектами приложения:
   - Статистика: Визуализация количества продаж.
   - Добавление товара: Форма с полями для ввода информации о новом товаре, включая название, цену и изображение.
   - Лист заказов: Список всех текущих заказов.

— Корзина: Интерфейс для просмотра выбранных товаров перед оформлением заказа.
— Магазин: Список доступных товаров с возможностью их просмотра, добавления в корзину.
— Кнопка выхода: Удобный способ выхода из системы администратором.

3. Страница покупателя. Эта страница предназначена для удобства покупок и управления личным кабинетом:
— Корзина: Интерфейс для просмотра выбранных товаров перед оформлением заказа.
— Магазин: Список доступных товаров с возможностью их просмотра, добавления в корзину.
— Кнопка удаления аккаунта: Предоставляет возможность полностью удалить свой аккаунт из системы.
— Кнопка выхода: Удобный способ выхода из системы покупателем.

**Инструменты для реализации**

**Язык разработки**

Для разработки прототипа CRM-системы был выбран C#, так как он является отличным решением благодаря нескольким ключевым преимуществам этого языка программирования. C# обладает мощными инструментами и библиотеками, которые идеально подходят для создания масштабируемых и надежных приложений.

**Пользовательский интерфейс приложения**

Для реализации пользовательского интерфейса (UI) в разработке прототипа CRM-системы был выбран Unity. Unity — это мощный движок для разработки игр, который также широко используется для создания интерактивных приложений и прототипов благодаря своим уникальным возможностям.

**База данных**

Для разработки и администрирования базы данных был выбран Navicat. Использование Navicat вместе с C# и Unity для разработки прототипа CRM-системы представляет собой сочетание инструментов, которое позволяет создавать комплексные и функциональные приложения. Navicat — это попу-

лярный инструмент для работы с базами данных, который поддерживает множество типов баз данных, включая MySQL, PostgreSQL, SQLite и многие другие.

В **третьем** разделе диплома проведена реализация прототипа.

**Реализация моделей (Model)**

В рамках реализации моделей данных, были рассмотрены примеры классов `User` и `Buyer`.

Класс `User`

1. `id`: Уникальный идентификатор пользователя, автоматически увеличивающийся при создании нового объекта.
2. `name`: Имя пользователя для идентификации в интерфейсе.
3. `password`: Пароль для аутентификации пользователя.

Были использованы геттеры и сеттеры для управления доступом к данным, обеспечивая инкапсуляцию.

Класс Buyer

`_name, _lastName, _surName, _phone, _dateOfBirth`,
`_registrationDate`: Поля для хранения информации о покупателе, включая имя, фамилию, отчество, номер телефона, дату рождения и дату регистрации. Каждое поле имеет соответствующий геттер и сеттер. Оба класса используются в контексте работы с базой данных.

**Реализация контроллеров (Controller)** Была рассмотрена реализация контроллеров на примере моделей пользователь (`User`) и покупатель (`Buyer`).

`UserController`

Класс `UserController` представляет собой компонент управления пользователями в приложении. Он использует базу данных для хранения информации о пользователях и управляет процессами входа в систему, добавления новых пользователей и удаления существующих. Описание его функционала:

1. Инициализация и Загрузка Пользователей. При запуске метода `Start` определяется путь к базе данных. Загружаются все пользователи из базы данных.
2. Вход в Систему
   — Метод `CheckUser` проверяет, существует ли пользователь с указан-

ным логином и паролем в списке `users`.

   — Метод `Loading` вызывается после ввода логина и пароля. Если пользователь успешно авторизован (с помощью `CheckUser`), отображается сообщение об успехе и вызывается событие `successfulLogin`. В противном случае вызывается событие `notSuccessfulLogin`.

3. Добавление Пользователя. Метод `AddUser` создает нового пользователя с заданным логином и паролем, добавляет его в список `users` и сохраняет в базе данных через `dbManager`.
4. Удаление Пользователя. Метод `DeleteUser` удаляет пользователя с указанным логином из списка `users` и базы данных через `dbManager`.

`BuyerController`

Класс `BuyerController` предназначен для управления информацией о покупателях. Этот класс отвечает за загрузку всех покупателей из базы данных, добавление новых покупателей и взаимодействие с базой данных через `DataBuyerManager`. Описание его функционала:

1. Инициализация и загрузка Покупателей. В методе `Start`, который вызывается при старте скрипта, определяется путь к файлу базы данных (`shop.bytes.db`) и создается экземпляр `DataBuyerManager`. Затем выполняется метод `GetAllUsers()` для загрузки всех покупателей из базы данных в список `buyers`.
2. Добавление покупателя. Метод `AddBuyer` собирает информацию о новом покупателе из полей ввода, создает новый объект `Buyer` с информацией, добавляет его в список `buyers` и сохраняет в базе данных через `dbManage`r.
3. Общая структура
   — `List of Buyers`: Список `buyers` используется для хранения всех текущих покупателей, что позволяет легко отслеживать всех клиентов системы.
   — `DataBuyerManager`: Класс `DataBuyerManager` отвечает за взаимодействие с базой данных, включая чтение и запись информации о покупателях.

**Реализация представлений (View)**

Был реализован **экран входа и регистрации**, являющийся точкой входа в систему для пользователей.

Проверка и отработка ошибок совпадений логина и пароля была реализована в контроллере пользователь (`UserController`). Функция запуска сессии администратора или покупателя была реализована в классе `AdminSession`.

Класс `AdminSession` является компонентом, предназначенным для управления видимостью элементов интерфейса администратора в зависимости от того, является ли текущий пользователь администратором. Этот класс использует компонент `UserController` для проверки прав администратора пользователя.

Была реализована **страница покупателя** предназначеная для удобства покупок и управления личным кабинетом.

Реализация магазина осуществлена с помощью классов `ShopItem` и `ShopUi`.

Класс `ShopItem` является компонентом, предназначенным для отображения товара в магазине и управления им. Этот класс позволяет пользователям покупать товары, добавляя их в корзину, и загружать аватары товаров асинхронно. Класс `ShopUi` является компонентом, предназначенным для управления пользовательским интерфейсом магазина в приложении. Этот класс использует префаб `PrefabItem` для создания визуальных элементов магазина, каждый из которых связан с определенным товаром через компонент `ShopItem`.

Реализация корзины осуществлена с помощью классов `ShoppingCartItem` и `ShoppingCartManager`.

Класс `ShoppingCartItem` предназначен для отображения элемента корзины покупок `(ShoppingCartItem)` в приложении. Этот класс использует компонент `Product` для управления данными товара, а также поля `Image`, `TextMeshProUGUI` для отображения информации о товаре и загрузки изображения. Класс `ShoppingCartManager` является компонентом, предназначенным для управления корзиной покупок в приложении. Этот класс использует список `products` для хранения всех доступных продуктов, а также префаб `ShoppingCartItem` для создания визуальных элементов корзины покупок.

Была реализована **страница администратора.**

Реализация окна добавления товара осуществлена классом `AddProduct`.

Класс `AddProduct` предназначен для добавления нового товара в систему через пользовательский интерфейс. Этот класс использует компонент `ProductController` для управления процессом добавления товара и поля ввода `(InputField)` для получения данных о товаре от пользователя.

Реализация списка заказов осуществлена классами `ItemList` и `ItemLIstManager`.

Класс `ItemList` предназначен для отображения списка элементов, таких как заказы `(Order)` и пользователи `(UserController)`. Этот класс использует компоненты `UserController` и `ProductController` для управления данными пользователей и товаров соответственно, поля `TextMeshProUGUI` для отображения информации о заказе.

Класс `ItemLIstManager` предназначен для управления списком элементов, таких как заказы `(Order)`, пользователи `(UserController)` и товары `(ProductController)`. Этот класс использует компоненты `OrderController`, `UserController` и `ProductController` для управления данными заказов, пользователей и товаров соответственно, а также префаб `ItemList` для создания экземпляров элементов списка.

Реализация статистики осуществлена классом `StatisticData`.

Класс `StatisticData` предназначенным для отображения статистических данных за последние семь дней в виде графика. Этот класс использует различные списки для хранения данных о датах, количествах заказов и элементах интерфейса, а также компонент `OrderController` для управления данными заказов.

Была разработана **SQL база данных**:

1. Для каждой сущности (User, Buyer, Order, Product) создана отдельная таблица.
2. В каждой таблице созданы столбцы, соответствующие атрибутам сущности. Например, для сущности User созданы столбцы UserId, Username, и Password.
3. Использованы первичные ключи (PRIMARY KEY) для уникального идентификатора каждой записи в таблице.
4. Использованы внешние ключи (FOREIGN KEY) для установления связей между таблицами.

5. Типы данных выбраны исходя из предполагаемых требований к данным.

В **заключении** приведены результаты бакалаврской работы.

## Основные результаты

1. Определены основные понятия, необходимые для описания автоматизированных систем учета взаимодействия с клиентами.
2. Изучены существующие типы автоматизированных систем учета взаимодействия с клиентами.
3. Разработано техническое задание и спроектирована архитектура приложения.
4. В ходе исследования реализован прототип CRM-системы, включающий базовые функциональные возможности, такие как управление заказами, ведение истории взаимодействия с клиентами, отображение статистики и т.д.
5. Разработанный прототип CRM-системы предлагает большую гибкость и контроль над данными, позволяя настраивать систему под конкретные потребности бизнеса. Также, благодаря возможности установки на собственном оборудовании, данный прототип обеспечивает надежность и безопасность хранения данных, что критически важно для многих организаций.