Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа по предмету «Проектирование информационных систем» на тему «Моделирование процессов с использованием методологии *IDEF3*»

Студент: Коршун Н.И.

ФИТ 4 курс 5 группа

Преподаватель: Якубенко К. Д.

Минск 2024

# 1 Постановка задач

Задачей проекта является моделирование процессов информационной системы на основе методологии IDEF3, разработка моделей бизнес-процессов и их декомпозиции. Модели должны отражать все функциональные требования, заявленные к информационной системе на основании предыдущих лабораторных работ (например, введение данных в форму, подтверждение количества сданных отходов пользователям).

Web-приложение должно быть асинхронным, взаимодействовать с базой данных и поддерживать разные платформы. UI реализуется на Next с асинхронностью, бизнес-логика, интерфейс и хранилище данных — максимально независимы для легкого расширения. Требуется разработать диаграмму вариантов использования (UML), логическую схему базы данных и структурную схему приложения. Язык — JavaScript, платформа — NodeJS, развертывание — через Docker.

Функционально web-приложение должно:

* обеспечивать возможность просмотра информации о различных моделях автомобилей (характеристики, фотографии);
* предоставлять функционал просмотра видеообзоров на автомобили;
* позволять пользователю сравнивать характеристики нескольких автомобилей;
* обеспечивать регистрацию и авторизацию пользователей с возможностью восстановления пароля через электронную почту
* поддерживать роли гостя, пользователя, администратора;
* для гостей доступен просмотр автомобилей и видеообзоров;
* для зарегистрированных пользователей доступен функционал сохранения избранных автомобилей и добавление автомобилей в список для сравнения;
* поддерживать функционал торговой площадки для продажи и покупки автомобилей с возможностью размещения объявлений, загрузки фотографий и контактов;
* администратор может добавлять и редактировать информацию об автомобилях, видеообзорах и управлять пользователями;
* поддерживать фильтрацию и поиск по различным параметрам автомобилей (марка, год выпуска, тип двигателя и т.д.).

Основные задачи приложения:

* разработать удобный интерфейс для поиска информации об автомобилях;
* предоставить функционал для просмотра видеообзоров на автомобили;
* создать систему сравнения характеристик автомобилей для более детального анализа.

# 2 Описание программных средств

Для выполнения лабораторной работы использовался ряд современных программных средств, каждое из которых играло ключевую роль в разработке и обеспечении функциональности системы.

Node.js (версия 18.17.1) был выбран в качестве основной платформы для разработки серверной части приложения. Это кроссплатформенное решение, поддерживающее работу на таких операционных системах, как Windows, Linux и macOS, что делает его крайне удобным для различных сред разработки и развертывания. Node.js предоставляет высокопроизводительную асинхронную модель обработки событий, которая позволяет эффективно управлять параллельными запросами пользователей, минимизируя задержки и обеспечивая плавную работу системы даже при высоких нагрузках. Особое внимание уделялось масштабируемости серверного решения, поскольку ожидается рост количества пользователей системы. Благодаря архитектуре Node.js, разработчики смогли реализовать систему, которая легко адаптируется к изменениям нагрузки, а также эффективно обрабатывает множество запросов на чтение и запись данных.

Важным преимуществом Node.js является его экосистема npm (Node Package Manager), которая предоставляет доступ к тысячам библиотек и инструментов, значительно ускоряя процесс разработки. Для данного проекта было использовано множество библиотек, что позволило минимизировать время на написание дополнительного кода, сосредоточив внимание на основной логике системы — обработке запросов, управлении авторизацией и взаимодействии с базой данных. Программный код на JavaScript и Node.js был оптимизирован с целью обеспечения высокой производительности и устойчивости системы в условиях масштабирования.

Данные о технологиях:

* Разработчик: OpenJS Foundation.
* Адрес загрузки: nodejs.org.
* Использовался для разработки серверной части системы, обеспечивая обработку запросов пользователей, управление авторизацией и взаимодействие с базой данных.
* Доступность: кроссплатформенная (Windows, Linux, macOS).

Next.js был выбран для создания клиентской части приложения, что позволяет пользователям взаимодействовать с системой через удобный веб-интерфейс. Next.js — это фреймворк для разработки динамических пользовательских интерфейсов, которая оптимизирует работу с компонентами и виртуальным DOM, что позволяет интерфейсу быстро реагировать на изменения данных без необходимости перезагрузки страницы. Благодаря гибкости Next.js, разработчикам удалось легко интегрировать асинхронные запросы, обеспечивая плавное взаимодействие пользователя с сервером. Данные о технологии:

* Разработчик: Facebook.
* Адрес загрузки: nextjs.org.
* Использовался для создания клиентской части приложения, асинхронного пользовательского интерфейса и взаимодействия с сервером.
* Доступность: кроссплатформенная (поддержка браузеров).

MySQL (версия 8.0.28) выполняет роль основной базы данных для хранения данных пользователей, статей, пунктов приема вторсырья и статистической информации. Этот реляционный менеджер баз данных был выбран за его высокую производительность, надежность и расширяемость. MySQL поддерживает мощные инструменты для обработки больших объемов данных, включая индексацию, параллельную обработку запросов и надежные транзакции. Это позволяет системе обрабатывать множество запросов на чтение и запись данных одновременно, обеспечивая стабильную работу даже при увеличении числа пользователей. Также стоит отметить, что MySQL кроссплатформенен и поддерживает различные операционные системы, что делает его гибким в использовании.

Наконец, Docker (версия 20.10) был использован для контейнеризации всех компонентов системы, что позволяет разворачивать приложение на различных платформах с минимальными усилиями. Docker обеспечивает изоляцию и независимость окружений, что упрощает процесс разработки, тестирования и развертывания приложения. Каждый компонент системы был упакован в отдельные контейнеры, что позволяет легко управлять зависимостями и масштабировать систему. Docker также предоставляет возможность горизонтального и вертикального масштабирования приложения, что особенно важно для поддержки высоких нагрузок в продакшене.

Таким образом, использование этих технологий обеспечило эффективную и масштабируемую разработку системы, соответствующую всем требованиям по функциональности и безопасности. Каждая технология сыграла свою уникальную роль в создании целостной архитектуры, которая способна адаптироваться к изменениям нагрузки и масштабироваться без ущерба для производительности. Благодаря контейнеризации с использованием Docker, разработчики смогли обеспечить гибкость развертывания системы, а также ее быструю настройку в различных окружениях. Кроссплатформенные возможности Node.js и MySQL, а также гибкость Next.js позволили обеспечить взаимодействие серверной и клиентской частей, при этом сохраняя высокий уровень производительности и отказоустойчивости.

# 3 Описание практического задания

Задачей является построение модели бизнес-процессов на основе методологии IDEF3. Необходимо:

* Построить IDEF0-модель на основе предыдущих этапов разработки.
* Разработать IDEF3-модель, представляющую функциональные блоки системы (например, обработку сообщений, авторизацию пользователей, управление чатами).
* Описать связи между блоками, потоки данных, хранилища и внешние объекты, которые участвуют в процессах.

Контекстная диаграмма добавления объявлений на торговую площадку представлена на рисунке 3.1.

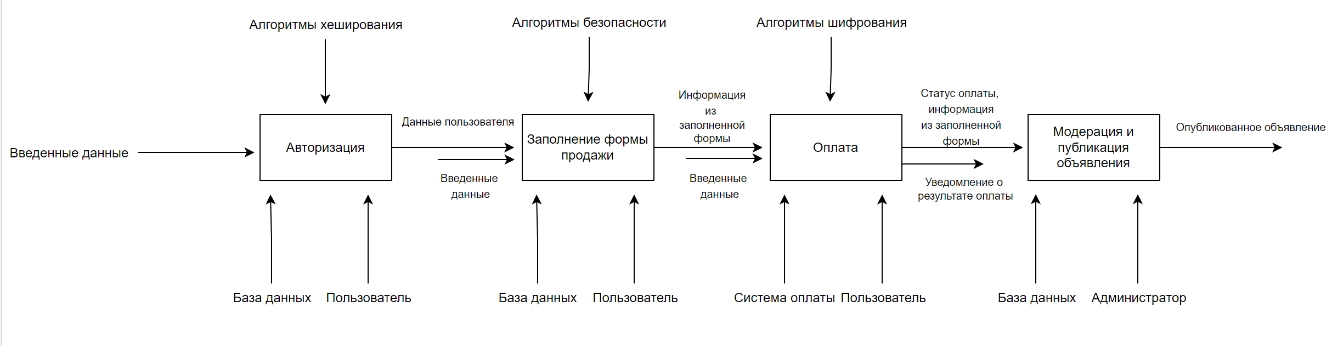


Рисунок 3.1 – Контекстная диаграмма управления торговой площадкой

Пример описания IDEF3 процесса:

* **Работа 1**: авторизация пользователя. Входные данные: заполненные поля формы (почта и пароль пользователя). Процесс: проверка введённых данных через базу данных пользователей. Выходные данные: сообщение о успешной авторизации.
* **Работа 2**: заполнение формы. Входные данные: параметры (марка, модель, год выпуска, пробег) и фотографии автомобиля. Процесс: проверка введенных данных через валидаторы. Выходные данные: сообщение о успешном заполнении формы.
* Работа 3: оплата. Входные данные: информация из заполненной формы. Процесс: проверка выполнения платежа. Выходные данные: сообщение об успешной оплате.
* Работа 4: модерация объявления. Входные данные: заполненная форма с данными и фотографиями автомобиля, отправленная пользователем. Процесс: проверка полученных данных на соответствие правилам площадки. Выходные данные: сообщение о успешном добавлении объявления.

Draw.io – это бесплатное онлайн-приложение для создания диаграмм и схем. Оно позволяет пользователям создавать диаграммы благодаря широкому набору инструментов и функций.

Работа заполнение формы представлена на диаграмме, представленной на рисунке 3.2.

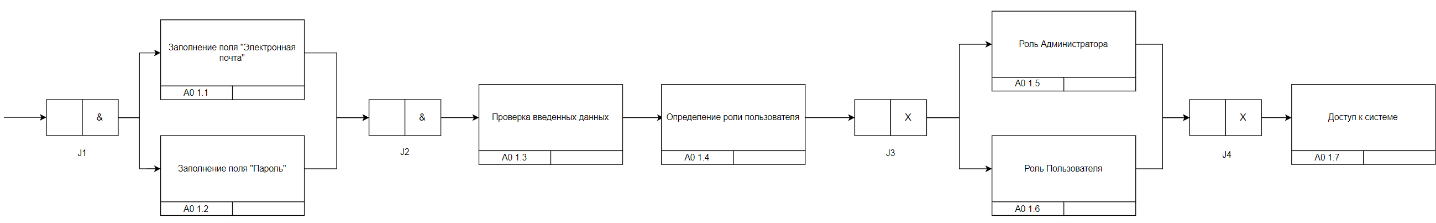


Рисунок 3.2 – Диаграмма процесса авторизации пользователя

Данная диаграмма отображает процесс авторизации пользователя. Входные потоки содержат поля формы (электронная почта и пароль пользователя). Потоки управления содержат поток валидации формы и поиска в базе данныз. Выходные потоки содержат информацию о результате авторизации. В качестве механизмов выступают пользователь веб-приложения и система.

Описание основных элементов:

* Заполнение поля формы – начальный этап, где пользователь вводит свои данные.
* Проверка введенных данных – система проверяет данные, чтобы определить, корректны ли они.
* В зависимости от результатов, система может вывести одно из следующих сообщений: данные прошли валидацию, данные не прошли валидацию.
* Поиск в базе данных – система проверяет в базе данных совпадают ли данные пользователя.
* В зависимости от результатов, система может вывести одно из следующих сообщений: успешная авторизация, ошибка авторизации.

Эта диаграмма подчеркивает логику работы системы, проверяя введенные данные для дальнейшего взаимодействия с приложением.

Диаграмма процесса заполнения формы представлена на рисунке 3.3.

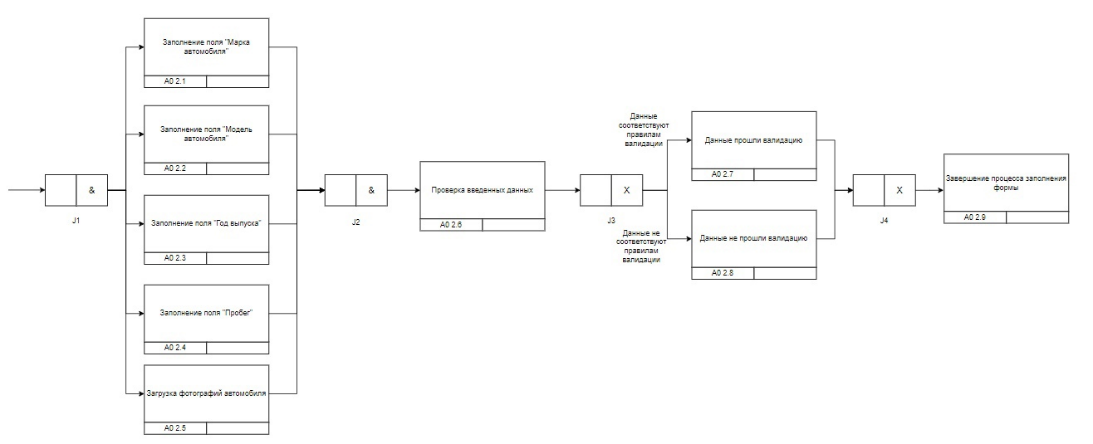


Рисунок 3.3 – Диаграмма процесса заполнения формы

На данной диаграмме описывается процесс заполнения формы продажи автомобиля.

Описание основных элементов:

* Заполнение поля формы – начальный этап, где пользователь вводит параметры автомобиля.
* Проверка введенных данных – система проверяет данные, чтобы определить, корректны ли они.

В зависимости от результатов, система может вывести одно из следующих сообщений: данные прошли валидацию, данные не прошли валидацию.

Диаграмма процесса оплаты представлена на рисунке 3.4.

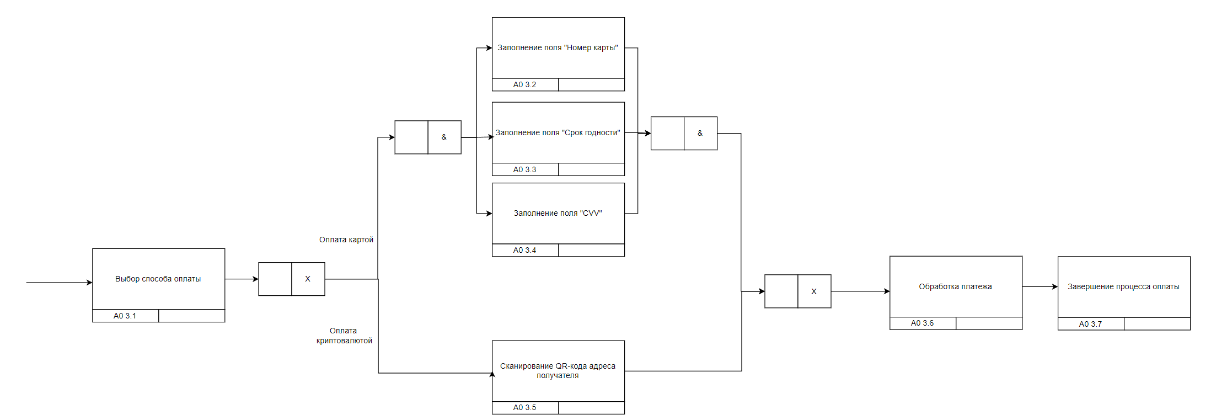


Рисунок 3.4 – Диаграмма процесса оплаты

На данной диаграмме описывается процесс оплаты добавления автомобиля на торговую площадку.

Описание основных элементов:

* Выбор способа оплаты – начальный этап, где пользователь выбирает каким методом он хочет совершить платеж.
* Заполнение полей – пользователь вводит данные карты, если он выбрал способ оплаты картой.
* Сканирование QR-кода адреса получателя – пользователь сканирует QR-код и переводит необходимую сумму по адресу получателя.
* Обработка платежа – система проверяет статус платежа.
* В зависимости от результатов, система может вывести одно из следующих сообщений: успешная оплата, не удалось оплатить.



Рисунок 3.5 – Диаграмма процесса модерации объявления

На рисунке 3.5 представлена диаграмма процесса модерации добавляемого объявления.

На данной диаграмме описывается процесс подтверждения введенных данных. Он состоит из шести функциональных блоков:

* Первый блок – получение заполненной формы объявления. Отвечает за получение данных из формы, когда пользователь ввел параметры автомобиля.
* Второй блок – проверка корректности информации. В этом блоке система анализирует введённые данные на предмет их корректности и соответствия требованиям площадки. Автоматические валидаторы проверяют, заполнены ли все обязательные поля, соответствуют ли данные форматам (например, числовые значения для пробега или года выпуска), а также нет ли явных ошибок (например, неправильно указаны марка или модель автомобиля).
* Третий блок – проверка на наличие запрещенного контента. На данном этапе проверяется, нет ли в предоставленных данных нарушений правил площадки. Особое внимание уделяется загруженным фотографиям, так как они могут содержать неподобающий или запрещённый контент. Система также может проверять текстовые описания на предмет недопустимой лексики или ложной информации.
* Четвертый блок – подтверждение добавления объявления. На этом этапе пользователю отправляется уведомление о том, что его объявление одобрено и будет опубликовано на площадке.
* Пятый блок – отказ в добавлении объявления. Пользователю отправляется уведомление с объяснением причин отказа и, при необходимости, рекомендациями по исправлению.
* Шестой блок –завершение процесса модерации объявления.

Диаграммы процессов продемонстрировали, как происходит обработка данных пользователя через различные функциональные блоки системы. Это позволило выявить основные потоки информации, взаимодействие с внешними объектами и механизмами, а также определить точки контроля данных, что в будущем облегчит поддержку и развитие системы. IDEF0-модель позволила структурировать основные функции системы и их взаимосвязи, а разработанные IDEF3-модели детализировали процессы, такие как заполнение формы и подтверждение данных.

# Ответы на теоретические вопросы

1. Дайте описание термину «процесс»?

Процесс ­– это последовательность действий или операций, направленных на достижение определенной цели или результата.

2. Какие основные методы входят в IDEF3?

[**Основные методы IDEF3**:](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,udsedgeshop,channelstable,ntpquery,devtoolsapi,udsinwin10,udsdlpconsent,udsfrontload,cspgrd,&shellsig=764419830fc0a1213ff1bc0bfd4f2796cbf4f76a&setlang=ru&darkschemeovr=1&udsps=0&udspp=0#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C0f10b50b-0d92-4ac0-88af-176327a5ff43)

* **Старшая связь (Precedence Link)**: показывает, что одна работа должна завершиться до начала другой.
* [**Отношения (Relational Link)**: используются для отображения связей между работами и объектами](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,udsedgeshop,channelstable,ntpquery,devtoolsapi,udsinwin10,udsdlpconsent,udsfrontload,cspgrd,&shellsig=764419830fc0a1213ff1bc0bfd4f2796cbf4f76a&setlang=ru&darkschemeovr=1&udsps=0&udspp=0#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C1%7C0f10b50b-0d92-4ac0-88af-176327a5ff43).
* **Потоки объектов (Object Flow)**: описывают использование объекта в нескольких работах.

3. Какие элементы являются центральными компонентами модели IDEF3?

**Центральные компоненты модели IDEF3**:

* [**Единицы работы (Unit of Work, UOW)**: основные элементы, представляющие действия или процессы](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,udsedgeshop,channelstable,ntpquery,devtoolsapi,udsinwin10,udsdlpconsent,udsfrontload,cspgrd,&shellsig=764419830fc0a1213ff1bc0bfd4f2796cbf4f76a&setlang=ru&darkschemeovr=1&udsps=0&udspp=0#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C3%7C0f10b50b-0d92-4ac0-88af-176327a5ff43).
* [**Связи (Links)**: показывают взаимоотношения между единицами работы](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,udsedgeshop,channelstable,ntpquery,devtoolsapi,udsinwin10,udsdlpconsent,udsfrontload,cspgrd,&shellsig=764419830fc0a1213ff1bc0bfd4f2796cbf4f76a&setlang=ru&darkschemeovr=1&udsps=0&udspp=0#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C4%7C0f10b50b-0d92-4ac0-88af-176327a5ff43).
* [**Перекрестки (Junctions)**: используются для отображения логики взаимодействия стрелок при слиянии и разветвлении](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,udsedgeshop,channelstable,ntpquery,devtoolsapi,udsinwin10,udsdlpconsent,udsfrontload,cspgrd,&shellsig=764419830fc0a1213ff1bc0bfd4f2796cbf4f76a&setlang=ru&darkschemeovr=1&udsps=0&udspp=0#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C5%7C0f10b50b-0d92-4ac0-88af-176327a5ff43).

4. В чём смысл использования перекрёстков в IDEF3?

Перекрестки отображают логику взаимодействия стрелок при слиянии и разветвлении, помогая визуализировать последовательность и условия выполнения процессов.

5. В чём отличия IDEF0 и IDEF3? Когда и как их целесообразно использовать?

[**Отличия IDEF0 и IDEF3**:](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,udsedgeshop,channelstable,ntpquery,devtoolsapi,udsinwin10,udsdlpconsent,udsfrontload,cspgrd,&shellsig=764419830fc0a1213ff1bc0bfd4f2796cbf4f76a&setlang=ru&darkschemeovr=1&udsps=0&udspp=0#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C2%7C0f10b50b-0d92-4ac0-88af-176327a5ff43)

**IDEF0**: используется для функционального моделирования, фокусируется на функциях и их взаимодействиях.

**IDEF3**: ориентирован на моделирование процессов и последовательностей действий.

IDEF0 целесообразно использовать для описания функций системы, а IDEF3 ­– для детального описания бизнес-процессов и их последовательностей.