|  |  |  |
| --- | --- | --- |
|  |  | УТВЕРЖДАЮ |
|  |  | Руководитель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.  М.П. |

ТЕХНИЧЕСКИЙ ПРОЕКТ

Приложения «конфигуратор ПК»

2024

Содержание

[1 Уточнение структур данных 2](#_Toc179812259)

[2 Формы и их пердставления 5](#_Toc179812260)

[3 Разработка алгоритма решения задачи 8](#_Toc179812261)

[4 Определения языка 15](#_Toc179812262)

[5 Структура программы и требования к техническим средствам 16](#_Toc179812263)

# Уточнение структур данных

У приложения конфигуратор ПК должна быть база данных. База данных изображена в логической ER диаграмме и физической ER диаграмме на рисунках 1 и 2.

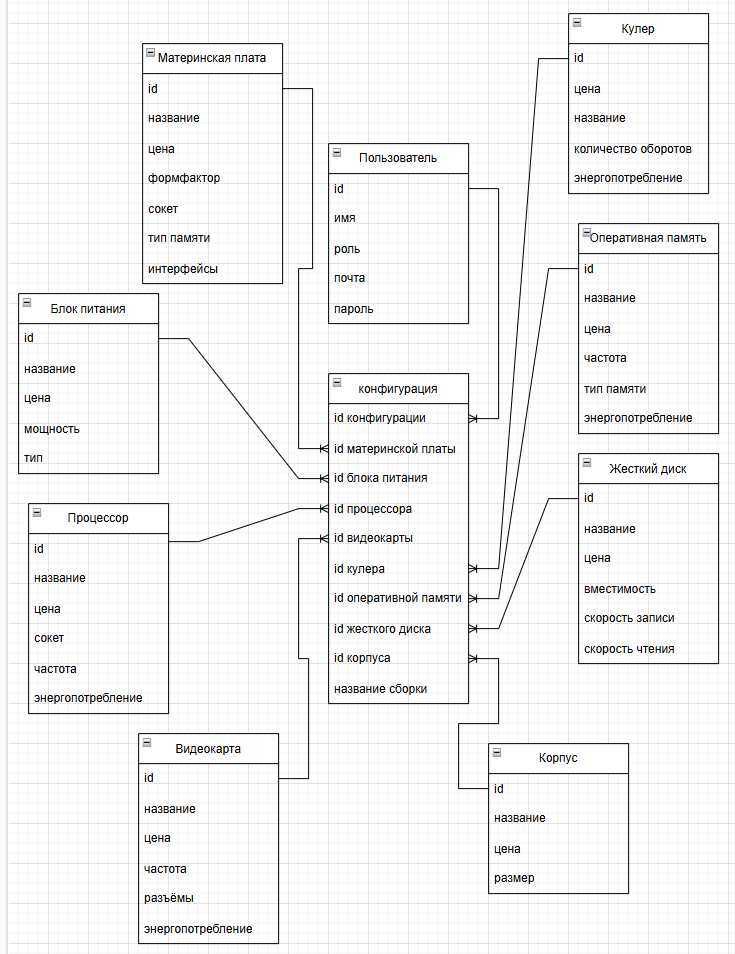


Рисунок 1 – Логическая ER диаграмма

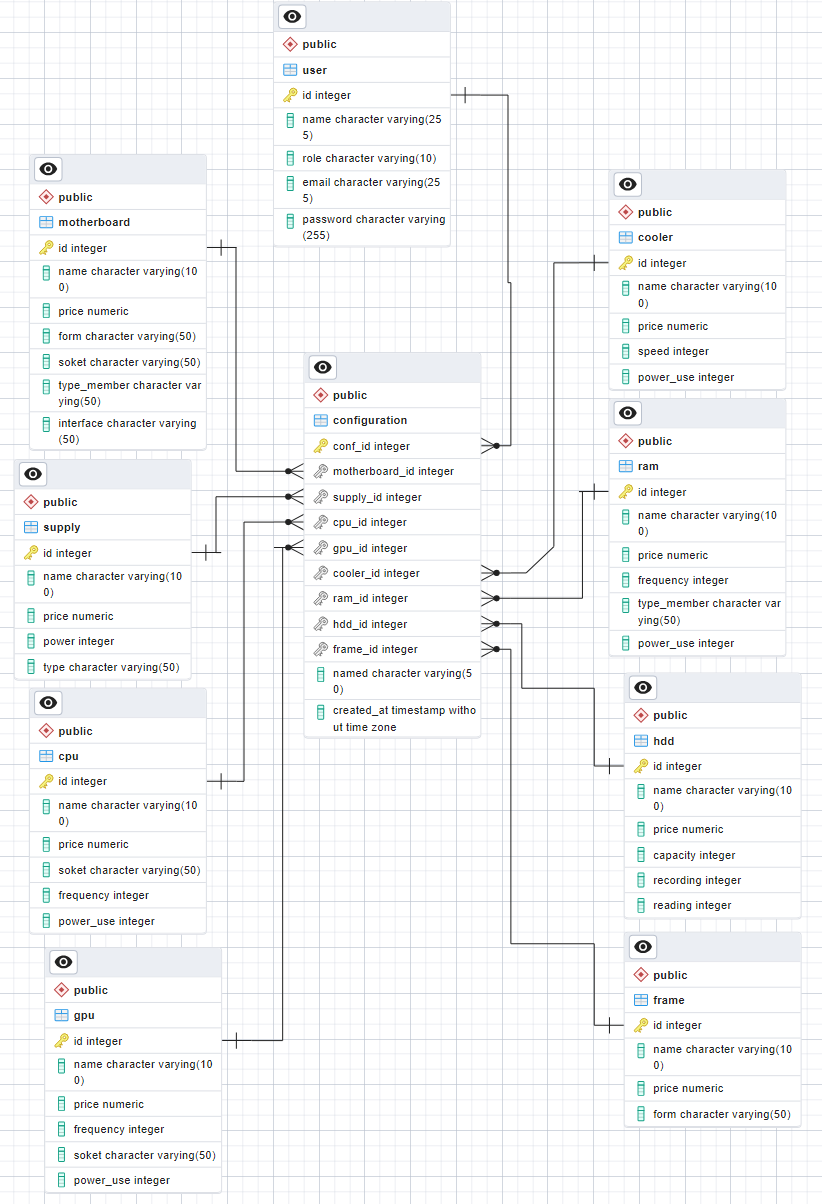


Рисунок 2 – Физическая ER диаграмма

На ER диаграмме показано 10 таблиц: «Материнская плата», «Блок питания», «Процессор», «Видеокарта», «Кулер», «Оперативная память», «Жесткий диск», «Корпус», «Конфигурация» и «Пользователь».

Таблица «Материнская плата»

* id: Уникальный идентификатор материнской платы.
* name: Название материнской платы.
* price: Стоимость материнской платы.
* form: Форм-фактор материнской платы (например, ATX, Micro-ATX).
* soket: Тип сокета, поддерживаемого материнской платой.
* Type\_member: Тип оперативной памяти, поддерживаемый материнской платой.
* interface: Список интерфейсов, доступных на материнской плате.

Таблица «Блок питания»

* id: Уникальный идентификатор блока питания.
* name: Название блока питания.
* price: Стоимость блока питания.
* power: Мощность блока питания в ваттах.
* type: Тип блока питания.

Таблица «Процессор»

* id: Уникальный идентификатор процессора.
* name: Название процессора.
* price: Стоимость процессора.
* soket: Тип сокета, поддерживаемый процессором.
* frequancy: Частота процессора.
* Power\_use: Потребляемая мощность процессора.

Таблица «Видеокарта»

* id: Уникальный идентификатор видеокарты.
* name: Название видеокарты.
* price: Стоимость видеокарты.
* frequancy: Частота видеокарты.
* soket: Список доступных разъемов на видеокарте.
* Power\_use: Потребляемая мощность видеокарты.

Таблица «Кулер»

* id: Уникальный идентификатор кулера.
* price: Стоимость кулера.
* name: Название кулера.
* speed: Количество оборотов кулера.
* Power\_use: Потребляемая мощность кулера.

Таблица «Оперативная память»

* id: Уникальный идентификатор оперативной памяти.
* name: Название оперативной памяти.
* price: Стоимость оперативной памяти.
* frequancy: Частота оперативной памяти.
* Type\_member: Тип оперативной памяти.
* Power\_use: Потребляемая мощность оперативной памяти.

Таблица «Жесткий диск»

* id: Уникальный идентификатор жесткого диска.
* name: Название жесткого диска.
* price: Стоимость жесткого диска.
* capacity: Объем жесткого диска.
* recording: Скорость записи данных на жесткий диск.
* reading: Скорость чтения данных с жесткого диска.

Таблица «Корпус»

* id: Уникальный идентификатор корпуса.
* name: Название корпуса.
* price: Стоимость корпуса.
* form: Размер корпуса.

Таблица «Конфигурация»

* conf\_id: Уникальный идентификатор конфигурации.
* Motherboard\_id: Ссылка на материнскую плату, используемую в конфигурации (внешний ключ к таблице «Материнская плата»).
* Supply\_id: Ссылка на блок питания, используемый в конфигурации (внешний ключ к таблице «Блок питания»).
* Cpu\_id: Ссылка на процессор, используемый в конфигурации (внешний ключ к таблице «Процессор»).
* Gpu\_id: Ссылка на видеокарту, используемую в конфигурации (внешний ключ к таблице «Видеокарта»).
* Cooler\_id: Ссылка на кулер, используемый в конфигурации (внешний ключ к таблице «Кулер»).
* Ram\_id: Ссылка на оперативную память, используемую в конфигурации (внешний ключ к таблице «Оперативная память»).
* Hdd\_id: Ссылка на жесткий диск, используемый в конфигурации (внешний ключ к таблице «Жесткий диск»).
* Frame\_id: Ссылка на корпус, используемый в конфигурации (внешний ключ к таблице «Корпус»).
* name: Название конкретной сборки или конфигурации.

Таблица «Пользователь»

* id: Уникальный идентификатор пользователя.
* name: имя пользователя
* role: роль (Пользователь или администратор).
* email: почта.
* Password: пароль.

# Формы и их пердставления

Прототип страницы «Создание конфигураций» на рисунке 3

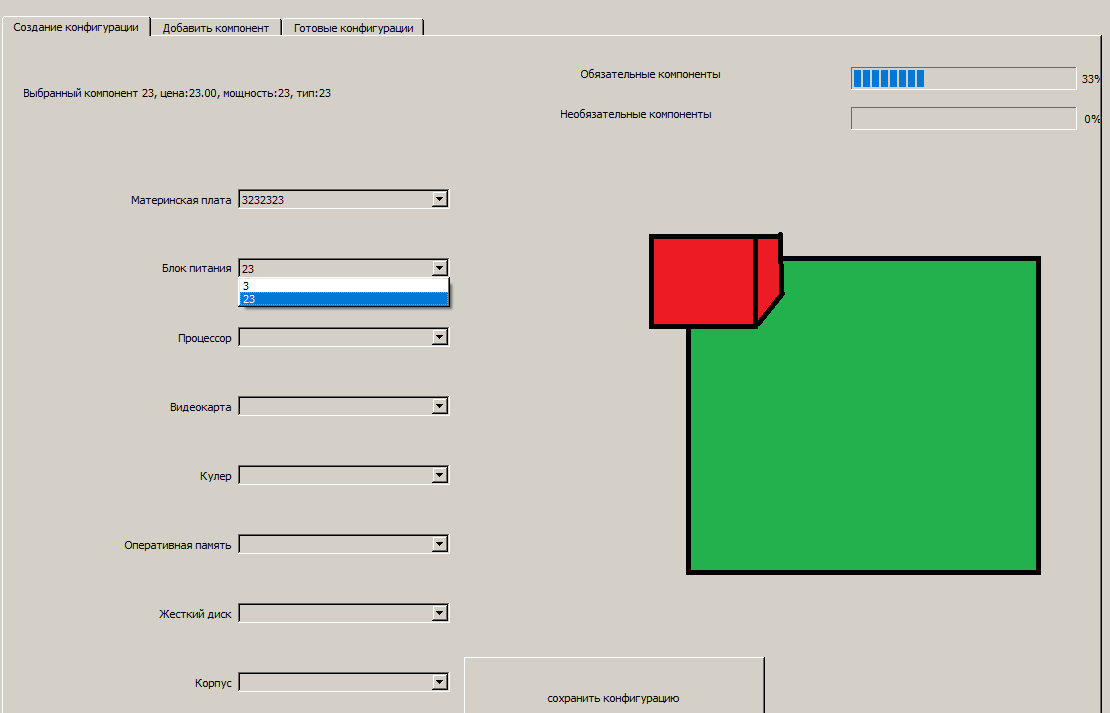


Рисунок 3 – Прототип страницы «Создание конфигураций»

На этом прототипе указано что пользователь может создавать конфигурации из добавленных комплектующих.

Прототип страницы «Создание компонентов» на рисунке 4

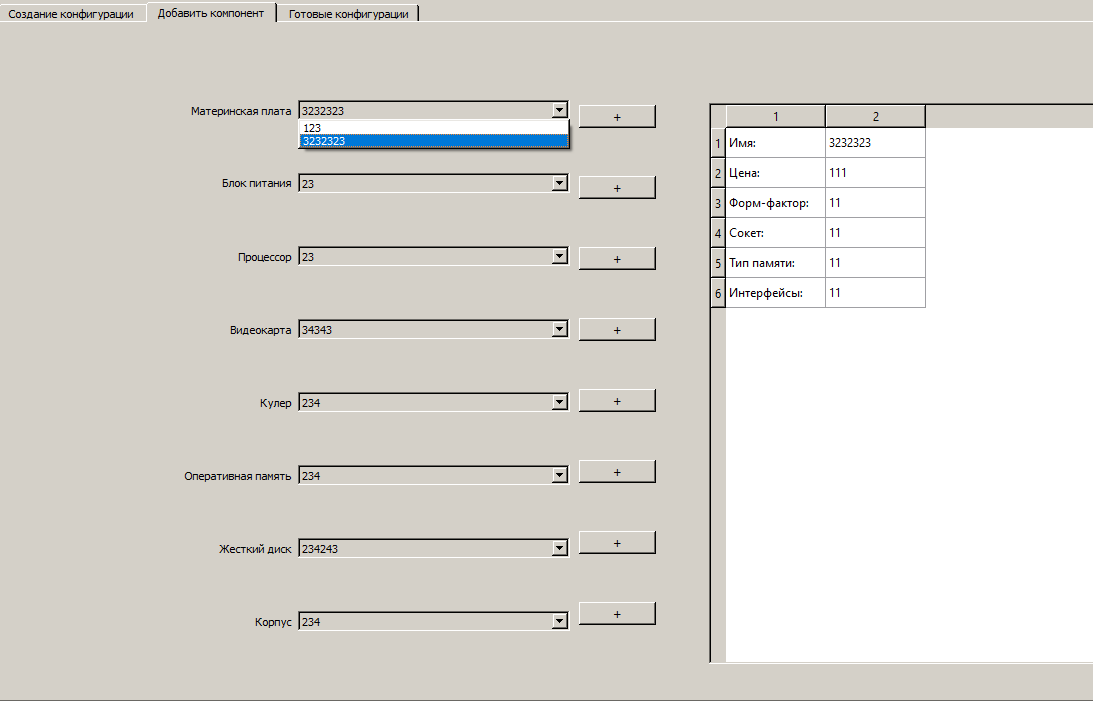


Рисунок 4 - Прототип страницы «Создание компонентов»

На этом рисунке показано что может делать администратор с комплектующими – удалять, добавлять и изменять.

Прототип страницы «Готовые конфигурации» на рисунке 5

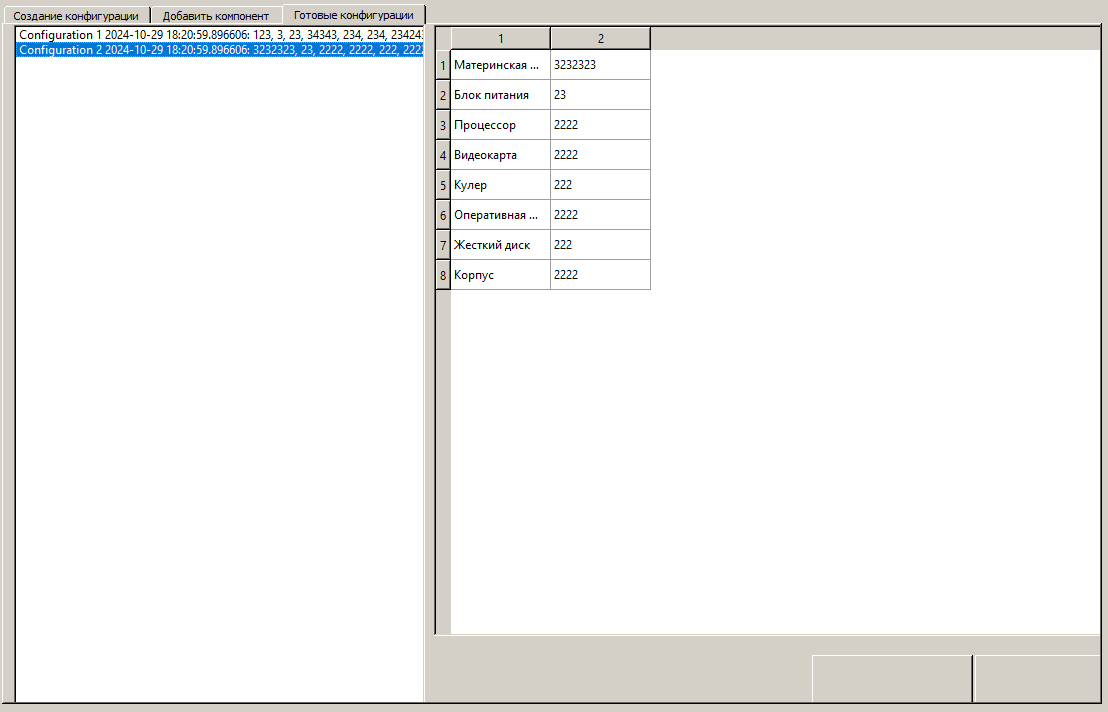


Рисунок 5 - Прототип страницы «Готовые конфигурации»

На этом прототипе указано что пользователь может изменять и удалять конфигурации которые он создал.

# Разработка алгоритма решения задачи

Для понимания как работает система в общем была создана диаграмма активности, в которой показаны последовательность всех алгоритмов и действий. Диаграмма активности изображена на рисунке 6.

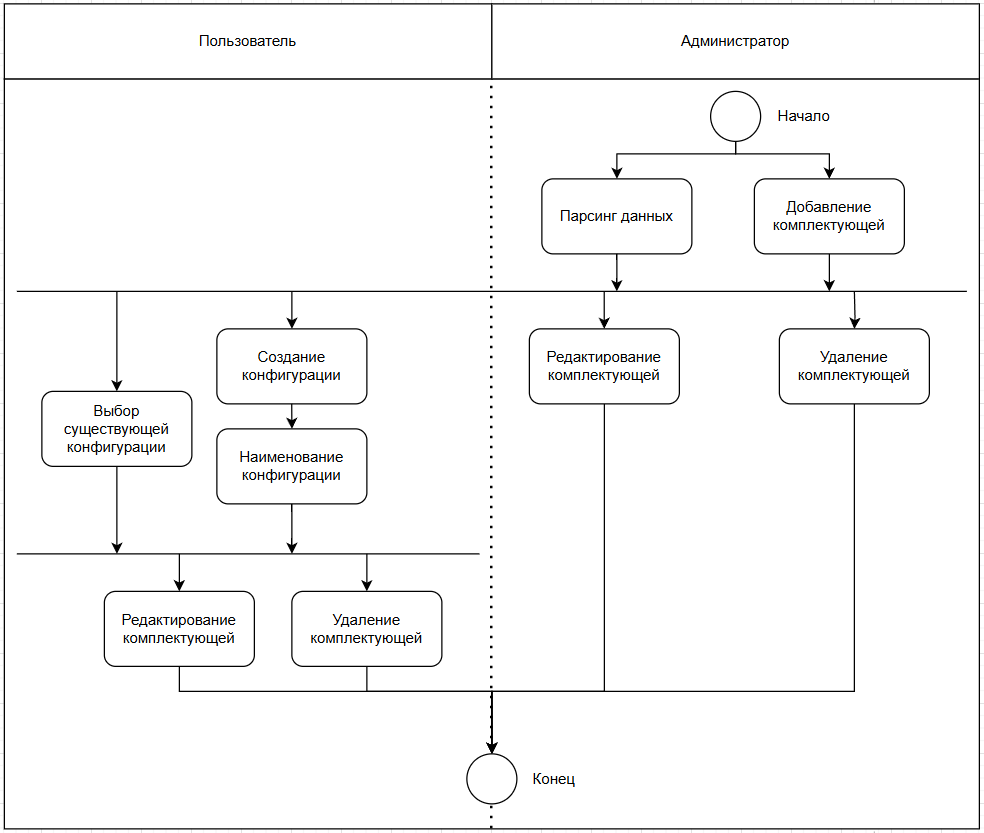


Рисунок 6 - Диаграмма активности.

Алгоритм «Добавление комплектующей**»**

Для лучшего понимания алгоритма «Добавление комплектующей» была создана блок схема, которая изображена на рисунке 7.

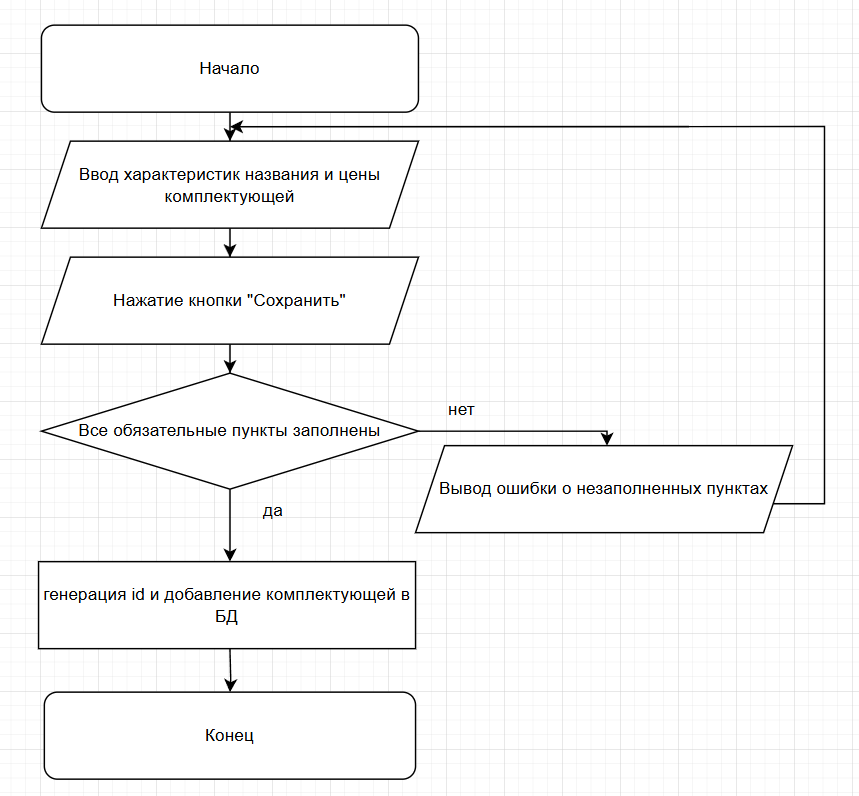


Рисунок 7 - Алгоритм «Добавление комплектующей**»**

Алгоритм «Изменение комплектующей**»**

Для лучшего понимания алгоритма «Изменение комплектующей» была создана блок схема, которая изображена на рисунке 8.

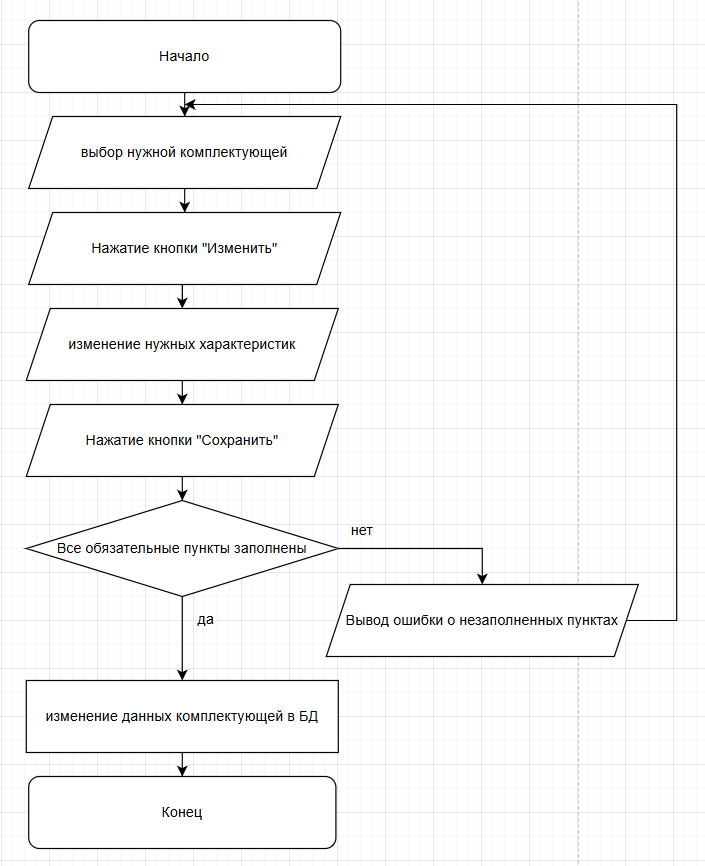


Рисунок 8 - Алгоритм «Изменение комплектующей**»**

Алгоритм «Удаление комплектующей**»**

Для лучшего понимания алгоритма «Удаление комплектующей» была создана блок схема, которая изображена на рисунке 9.

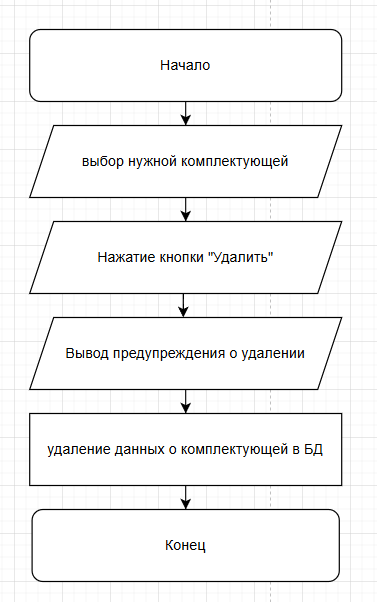


Рисунок 9 - Алгоритм «Удаление комплектующей**»**

Алгоритм «Создание конфигурации**»**

Для лучшего понимания алгоритма «Создание конфигурации» была создана блок схема, которая изображена на рисунке 10.

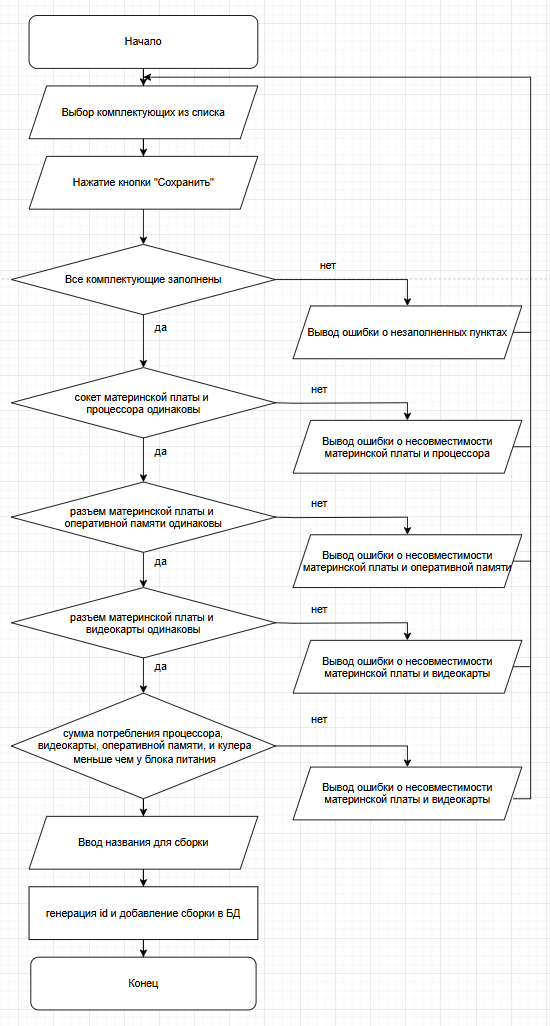


Рисунок 10 - Алгоритм «Создание конфигурации**»**

Алгоритм «Изменение конфигурации**»**

Для лучшего понимания алгоритма «Изменение конфигурации» была создана блок схема, которая изображена на рисунке 11.

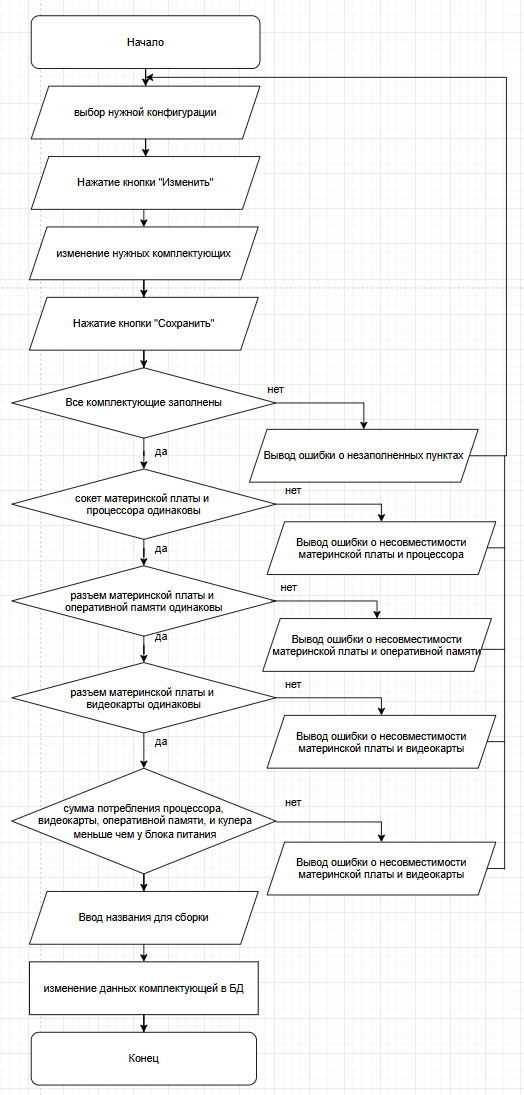


Рисунок 11 - Алгоритм «Изменение конфигурации**»**

Алгоритм «Удаление конфигурации**»**

Для лучшего понимания алгоритма «Удаление конфигурации» была создана блок схема, которая изображена на рисунке 12.

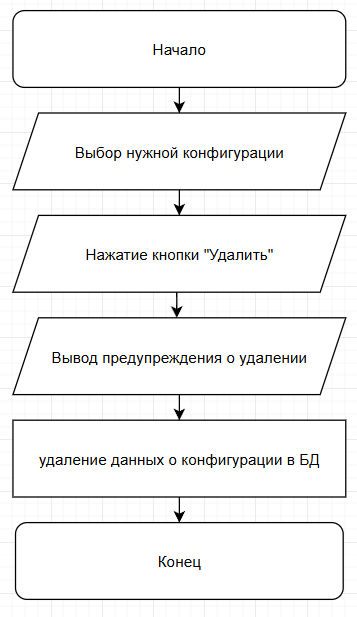


Рисунок 12 - Алгоритм «Удаление конфигурации**»**

Для понимания структуры проекта, была нарисована диаграмма Idef0. Диаграмма Idef0 изображена на рисунке 13 для пользователя и на рисунке 14 для администратора.

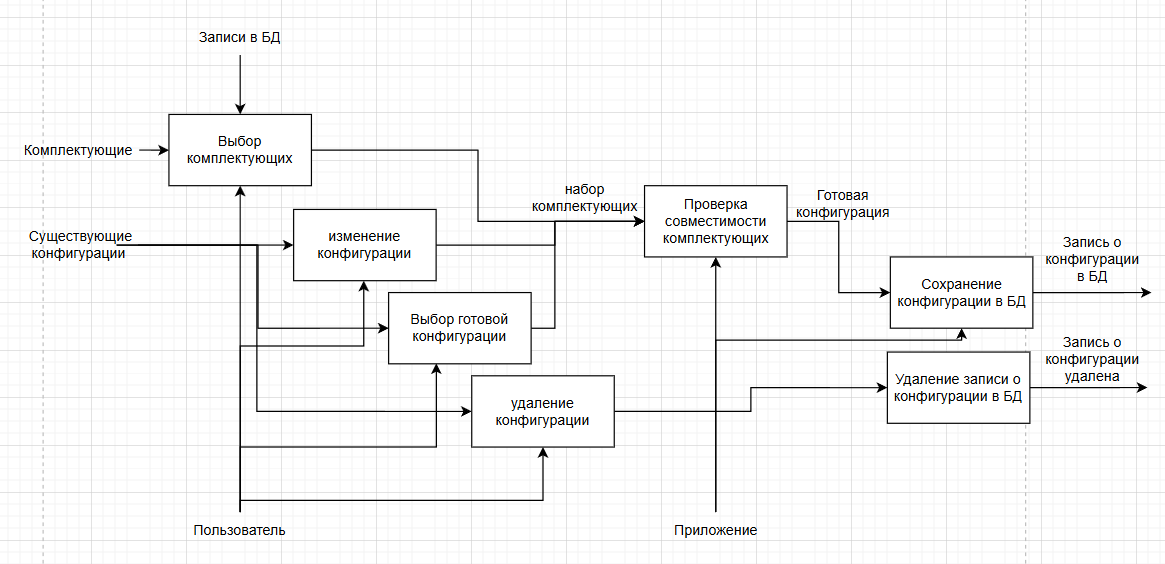


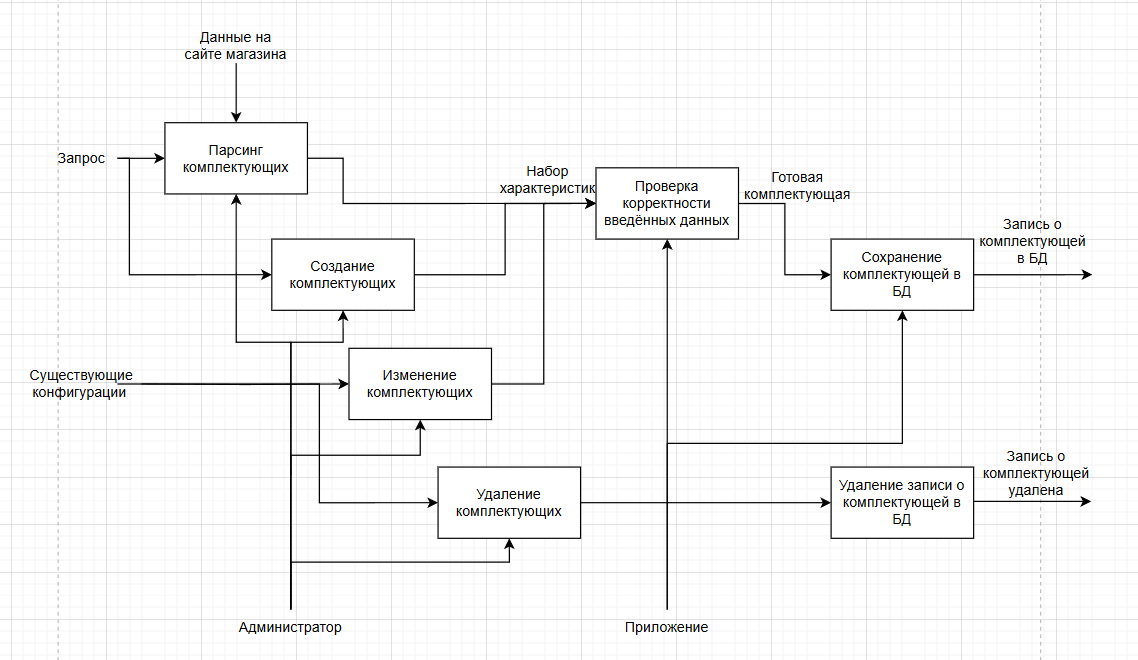
Рисунок 13 – idef0 для пользователя

Рисунок 14 – idef0 для администратора

Так же для понимания как реагирует система был составлен пользовательский сценарий, который указан в таблице 1 и 2.

Таблица 1 – Пользовательские сценарий (для администратора)

|  |  |  |
| --- | --- | --- |
| № функции | Администратор | Система |
| 1 | Добавление комплектующей | 1. Отображает форму для ввода данных о комплектующей и выбора настроек доступа.  2. После ввода данных и нажатия кнопки "Добавить", создает новую комплектующую с указанными параметрами и заносит ее в БД.  3. Обновляет интерфейс программы |
| 2 | Изменение комплектующей | 1. Подставляет характеристики выбранной комплектующей в таблицу для изменения характеристик.  2. После ввода данных и нажатия кнопки "Добавить", обновляет данные о комплектующей в БД.  3. Обновляет интерфейс программы |
| 3 | Удаление комплектующей | 1. Выводит предупреждение о удалении комплектующей  2. Удаляет запись о комплектующей в БД  3. Обновляет интерфейс программы |
| 4 | Парсинг данных | 1. Запускает скрипт на Selenium 2. Заносит запаршенные данные в БД 3. Обновляет интерфейс программы |

|  |  |  |
| --- | --- | --- |
| № функции | Администратор | Система |
| 1 | Создание конфигурации | 1. Отображает форму для создания конфигурации с ячейками для выбора комплектующих.  2. После введения данных проверяет совместимость компонентов и заполнены ли все обязательные поля.  3. Обновляет интерфейс программы и создает запись в БД. |
| 2 | Изменение конфигурации | 1.Отображает форму с сохраненными конфигурациями.  2. Перенаправляет пользователя на форму для создания конфигурации.  3. После введения данных проверяет совместимость компонентов и заполнены ли все обязательные поля.  4. Обновляет интерфейс программы и обновляет запись в БД. |
| 3 | Удаление конфигурации | 1. Выводит предупреждение о удалении конфигурации  2. Удаляет запись о конфигурации в БД  3. Обновляет интерфейс программы |
| 4 | Выбор готовой конфигурации | 1. Отображает готовые пресеты.   2. При нажатии на них перенаправляет пользователя на форму для создания конфигурации.  3. После введения данных проверяет совместимость компонентов и заполнены ли все обязательные поля.  4. Обновляет интерфейс программы и обновляет запись в БД. |

Таблица 2 – Пользовательские сценарий (для пользователя)

# Определения языка

Для реализации проекта "Онлайн доска", будут использоваться следующие языки программирования и технологии:

* **Vue.js (JavaScript framework)** – фреймворк, предназначенный для создания динамических пользовательских интерфейсов. Vue.js будет использоваться для разработки каркаса и логики интерфейса конфигуратора, включая работу с компонентами ПК и сборками, а также обеспечит адаптивность и удобство использования.
* **Flask**: легковесный фреймворк для создания API-интерфейсов, который отлично подходит для небольших проектов.
* **psycopg2 (PostgreSQL driver)** – драйвер для работы с базой данных PostgreSQL из Python-приложения. psycopg2 используется для установки соединения с базой данных, выполнения запросов и обработки транзакций.
* **PostgreSQL (Relational Database Management System)** – реляционная СУБД, используемая для хранения данных о компонентах ПК, конфигурациях и результатах парсинга. PostgreSQL обеспечит устойчивое хранение данных и эффективные запросы к большим объемам информации, с возможностью применения сложных фильтров и запросов.
* **Selenium (Automated Browser Control)** – инструмент для автоматизации действий в браузере, который необходим для обхода защиты от ботов на веб-страницах, динамически загружающих контент. Selenium позволяет симулировать действия пользователя, обходить JavaScript-защиту и получать нужные данные.

# Структура программы и требования к техническим средствам

Функциональные требования:

* Регистрация и авторизация пользователей
* Каталогизирование сборок (создание, редактирование, удаление)
* Каталогизирование комплектующих (создание, редактирование, удаление)
* Парсинг данных о комплектующих с интернет магазинов
* Возможность выбрать сборку из готовых вариантов

Нефункциональные требования:

* Удобство использования: интуитивно понятный и простой интерфейс, адаптивный дизайн для различных устройств
* Надежность: система должна быть стабильной и доступной большую часть времени