Автономная некоммерческая образовательная организация высшего образования Центросоюза Российской Федерации

СИБИРСКИЙ УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ

Кафедра информатики

**ОТЧЕТ О ПРОХОЖДЕНИИ ТЕХНОЛОГИЧЕСКОЙ**

**(ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКИ**

Место прохождения практики: *Акционерное общество «ТИОН Умный микроклимат»*

Студента очной формы обучения

Факультета Экономики и управления

Специальности *09.03.03 Прикладная информатика*

4 курса, группы ПИБ-11

*Колесников Сергей Андреевич*

Руководитель практики

*преподаватель(практик)*

*Блок Иван Николаевич*

Оценка после защиты

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сроки прохождения практики:

с 2.04.2025 по 16.05.2025

Новосибирск 2025

автономная некоммерческая образовательная организация высшего образования

Центросоюза Российской Федерации  
«Сибирский университет потребительской кооперации»

**РАБОЧИЙ ГРАФИК И ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

на технологическую (проектно-технологическую) практику

Ф.И.О. обучающегося Колесников Сергей Андреевич

Факультет Экономики и управления

Группа ПИБ-11

Кафедра Информатики

Направление подготовки: 09.03.03 Прикладная информатика

Профиль (направленность) Прикладная информатика в информационной сфере

1. Сроки практики с 2.04 по 16.05.2025 г.
2. Место прохождения практики Акционерное общество «ТИОН Умный микроклимат», г. Новосибирск, ул. Инженерная 20
3. Сроки сдачи студентом отчёта 16.05.2025 г.

| **№ п/п** | **Наименование работ** | **Срок** |
| --- | --- | --- |
| 1 | Инструктаж по охране труда, технике безопасности, соблюдению производственной санитарии и гигиены труда, а также правилам внутреннего распорядка. |  |
| 2 | Проведение анализа деятельности организации. Согласование с руководителем практики задания, постановки цели и задач практики. |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 | Обобщение и систематизация полученной информации и результатов анализа данных. Оформление отчета по результатам практики. |  |
| 12 | Подготовка презентации к публичной защите отчета. |  |

Руководитель практики Блок И.Н. 2.04.2025

Задание принял к исполнению Колесников С.A. 2.04.2025

Соруководитель практики от   
организации Михайлова Е.А. 2.04.2025

*.*

**ДНЕВНИК ПРАКТИКИ**

технологическая (проектно-технологическая) практика

Ф.И.О. обучающегося Колесников Сергей Андреевич

Факультет Экономики и управления

Группа ПИБ-11

Кафедра Информатики

Направление подготовки: 09.03.03 Прикладная информатика

Профиль (направленность) Прикладная информатика в информационной сфере

Место прохождения практики Акционерное общество «ТИОН Умный микроклимат», г. Новосибирск, ул. Инженерная 20

Сроки практики: с 2.04 по 16.05.2025 г.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Содержание этапа практики  (в соответствии с рабочим графиком и индивидуальным заданием) | Сроки  выполнения | Отметка руководителя практики о выполнении (выполнено/ не выполнено) |
|  | Инструктаж по охране труда, технике безопасности, соблюдению производственной санитарии и гигиены труда, а также правилам внутреннего распорядка. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Выписка из журнала вводного инструктажа Акционерное общество «ТИОН Умный микроклимат», г. Новосибирск, ул. Инженерная 20

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | ФИО инструктирующего | Подпись  инструктирующего | Подпись инструктируемого |
|  |  |  |  |

Специалист по развитию  
авторизованных сервисных центров Михайлова Е.А.

Директор Яковчук И.В.

Договор сдан в отдел

практической подготовки

и содействия трудоустройству \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отзыв**о работе практиканта

Обучающийся АНОО ВО Центросоюза Российской Федерации «Сибирский университет потребительской кооперации» Колесников Сергей Андреевич факультета экономики и управления, 4 курса, направления подготовки: 09.03.03 *Прикладная информатика*, профиль *Прикладная информатика в информационной сфере*, проходил технологическую (проектно-технологическую) практикув Акционерном обществе «ТИОН Умный микроклимат», г. Новосибирск, ул. Инженерная 20

В процессе прохождения практики и выполнения заданий обучающийся придерживался (не придерживался) рабочего графика и индивидуального задания, ответственно (недостаточно ответственно, неответственно) относился к поставленным задачам, последовательно (не системно) решал их.

| Критерии оценивания | | Критерии оценки | | | |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 |
| **1** | Уровень теоретической подготовленности студента к прохождению практической подготовки |  | + |  |  |
| **2** | Уровень практической подготовленности студента к прохождению практической подготовки |  | + |  |  |
| **3** | Умение правильно определять и эффективно решать основные задачи | +  \_ |  |  |  |
| **4** | Самостоятельность при выполнении задания | + |  |  |  |
| **5** | Уровень выполнения индивидуальных заданий | + |  |  |  |
| **6** | Трудовая дисциплина | + |  |  |  |

Практика оценивается (по 5-балльной шкале) ОТЛИЧНО.

Специалист по развитию  
авторизованных сервисных центров Михайлова Е.А.

Директор Яковчук И.В.

**Оглавление**

[ВВЕДЕНИЕ 6](#_Toc196946579)

[1 Анализ деятельности организации 7](#_Toc196946580)

[1.1 Общие сведения о компании ТИОН 7](#_Toc196946581)

[1.2 Анализ организационной структуры 7](#_Toc196946582)

[1.3 Анализ бизнес-процессов компании, связанных с конфигурацией продукции 8](#_Toc196946583)

[1.4 Анализ целевой аудитории конфигуратора 10](#_Toc196946584)

[1.5 Выводы по главе 11](#_Toc196946585)

[2 Выбор инструментальных средств разработки информационной системы 12](#_Toc196946586)

[2.1 Введение 12](#_Toc196946587)

[2.2 Выбор серверного фреймворка — ASP.NET Core Web API 13](#_Toc196946588)

[2.3 Обоснование выбора ORM — Entity Framework (EF) Core 15](#_Toc196946589)

[2.4 Выбор клиентского фреймворка — React 17](#_Toc196946590)

[2.5 Выбор системы управления базами данных — PostgreSQL 19](#_Toc196946591)

[2.6 Использование Docker для разработки и развертывания 21](#_Toc196946592)

[2.7 Вывод по главе 24](#_Toc196946593)

[3 Проектирование и разработка интерактивного конфигуратора 25](#_Toc196946594)

[3.1 Анализ требований к конфигуратору 25](#_Toc196946595)

[3.1.1 Функциональные требования 25](#_Toc196946596)

[3.1.2 Нефункциональные требования 26](#_Toc196946597)

[3.1.3 Технические требования 27](#_Toc196946598)

[3.2 Проектирование архитектуры системы 27](#_Toc196946599)

[3.3 Проектирование базы данных 29](#_Toc196946600)

[3.3.1 Проектирование инфологической модели БД 29](#_Toc196946601)

[3.3.2 Проектирование даталогической модели БД 30](#_Toc196946602)

[Заключение 32](#_Toc196946603)

[Список литературы 33](#_Toc196946604)

ВВЕДЕНИЕ

Целью данной технологической (проектно-технологической) практики является проектирование интерактивного конфигуратора для компании ТИОН, занимающейся производством и продажей систем вентиляции и очистки воздуха. Данный конфигуратор призван оптимизировать процесс подбора оборудования для клиентов, повысить точность формирования заказов и улучшить пользовательский опыт.

Задачи практики:

1. Изучение деятельности компании ТИОН и анализ существующих бизнес-процессов, связанных с конфигурацией продукции.
2. Сбор и анализ требований к интерактивному конфигуратору.
3. Проектирование архитектуры информационной системы, включая frontend, backend и базу данных.
4. Разработка функциональности конфигуратора, включающей управление каталогом продукции, настройку параметров оборудования, формирование спецификации и экспорт данных.
5. Тестирование разработанной системы и выявление возможных ошибок.
6. Документирование процесса разработки и результатов практики.

Актуальность проекта обусловлена необходимостью автоматизации и оптимизации процесса подбора оборудования ТИОН. В настоящее время данный процесс может быть трудоемким и подверженным ошибкам, особенно при работе со сложными конфигурациями. Интерактивный конфигуратор позволит клиентам и менеджерам по продажам самостоятельно подбирать оптимальное оборудование, сокращая время на формирование заказов и повышая их точность. Внедрение конфигуратора позволит повысить лояльность клиентов и увеличить объемы продаж.

1 Анализ деятельности организации

1.1 Общие сведения о компании ТИОН

ТИОН – это группа научно-исследовательских, производственных и торговых подразделений, обеспечивающих полный цикл создания современных высокотехнологичных продуктов в области умной и энергоэффективной вентиляции, очистки и обеззараживания воздуха.

Компания «ТИОН» является одним из ведущих российских предприятий в области разработки, производства и внедрения систем увлажнения и очистки воздуха. Основанная в 2006 году, организация успешно заняла свою нишу на рынке климатического оборудования, предлагая инновационные решения в сегменте бытового и промышленного увлажнения воздуха. Основным направлением деятельности компании является производство аппаратуры, способствующей созданию комфортных и здоровых микроклиматических условий в жилых, коммерческих и промышленных помещениях.

Основной целью деятельности компании «ТИОН» является разработка передовых технологий, обеспечивающих высокое качество воздуха и оптимальные параметры влажности, что напрямую влияет на здоровье и самочувствие пользователей. В рамках стратегии развития компания активно инвестирует в научно-исследовательские работы, модернизацию производства и расширение ассортимента продукции.

1.2 Анализ организационной структуры

Организационная структура компании построена по функциональному принципу и включает в себя несколько основных подразделений: производственное, маркетинговое, научно-исследовательское и сбытовое. Производственный отдел отвечает за изготовление и контроль качества выпускаемой продукции. Научно-исследовательское подразделение разрабатывает новые технологические решения и совершенствует существующие модели оборудования. Маркетинговая служба занимается анализом рынка, продвижением продукции и взаимодействием с клиентами. Сбытовой отдел осуществляет поставки и логистику.

Четкое распределение функций в организации способствует эффективному управлению процессами, оперативному решению производственных задач и своевременному реагированию на изменения рыночной среды.

1.3 Анализ бизнес-процессов компании, связанных с конфигурацией продукции

В настоящее время процесс подбора оборудования ТИОН для клиентов осуществляется менеджерами по продажам вручную, с использованием каталогов продукции. Часто требуется консультация с техническими специалистами для проверки совместимости оборудования.

Основные этапы этого процесса:

1. Сбор требований от клиента: Менеджер по продажам связывается с клиентом (по телефону, электронной почте или лично) и выясняет его потребности в системе вентиляции и очистки воздуха. Собирается информация о типе помещения, его площади, количестве людей, находящихся в помещении, и других факторах, влияющих на выбор оборудования.
2. Предварительный подбор оборудования: Менеджер, основываясь на полученных требованиях, подбирает подходящее оборудование из каталога продукции. Используются таблицы технических характеристик и рекомендации по применению оборудования в различных типах помещений.
3. Согласование конфигурации с техническими специалистами: Предварительно подобранная конфигурация согласовывается с техническими специалистами для проверки совместимости оборудования и соответствия требованиям клиента.
4. Составление коммерческого предложения: На основе согласованной конфигурации составляется коммерческое предложение, в котором указывается перечень оборудования, его стоимость и сроки поставки.
5. Согласование коммерческого предложения с клиентом: Коммерческое предложение направляется клиенту для согласования. В процессе согласования могут вноситься изменения в конфигурацию оборудования.
6. Оформление заказа: После согласования коммерческого предложения оформляется заказ на поставку оборудования.

Проблемы и недостатки существующего процесса:

* Длительное время подбора оборудования: Процесс подбора оборудования может занимать значительное время, особенно при работе со сложными конфигурациями и большими объемами заказов.
* Высокая вероятность ошибок при ручном подборе оборудования: Ручной подбор оборудования подвержен ошибкам, связанным с человеческим фактором, таким как невнимательность и недостаточная квалификация менеджеров по продажам.
* Зависимость от квалификации менеджера по продажам: Качество подбора оборудования во многом зависит от квалификации и опыта менеджера по продажам.
* Неудобство для клиентов, желающих самостоятельно подобрать оборудование: Клиенты, желающие самостоятельно подобрать оборудование, сталкиваются с трудностями при изучении каталогов продукции и технических характеристик.
* Сложность визуализации конечного результата: Клиентам сложно представить, как будет выглядеть система вентиляции и очистки воздуха в их помещении на основе предоставленных технических данных.

1.4 Анализ целевой аудитории конфигуратора

Целевой аудиторией интерактивного конфигуратора являются:

Менеджеры по продажам: Им нужен инструмент, который позволит быстро и точно подбирать оборудование, формировать коммерческие предложения и сокращать время на обслуживание клиентов. Они должны иметь возможность легко ориентироваться в каталоге продукции, настраивать параметры оборудования и получать информацию о его стоимости и наличии на складе.

Технические специалисты: Им нужен инструмент, который позволит проверять правильность конфигурации и избегать технических ошибок. Они должны иметь возможность анализировать технические характеристики оборудования, проверять совместимость компонентов и получать информацию о возможных ограничениях.

Конечные клиенты (дилеры, строительные компании, частные лица): Им нужен инструмент, который позволит самостоятельно подобрать оборудование, соответствующее их потребностям, и получить информацию о его характеристиках и стоимости. Они должны иметь возможность визуализировать конечный результат, сравнивать различные варианты конфигурации и получать консультации от специалистов компании.

Уровень технической подготовки целевой аудитории:

Уровень технической подготовки целевой аудитории варьируется в широких пределах. Менеджеры по продажам и конечные клиенты могут не обладать глубокими техническими знаниями, поэтому конфигуратор должен быть интуитивно понятным и простым в использовании. Технические специалисты, напротив, обладают высоким уровнем технической подготовки и нуждаются в инструменте, который предоставляет им доступ к детальной информации о продуктах и их характеристиках.

1.5 Выводы по главе

Компания ТИОН является одним из ведущих российских производителей систем вентиляции и очистки воздуха. Процесс подбора оборудования в настоящее время является трудоемким и подверженным ошибкам. Разрабатываемый интерактивный конфигуратор призван автоматизировать и оптимизировать этот процесс, повысить точность формирования заказов и улучшить пользовательский опыт. Внедрение конфигуратора позволит сократить время на подбор оборудования, снизить количество ошибок, повысить лояльность клиентов и увеличить объемы продаж. Конфигуратор должен быть удобным в использовании для пользователей с разным уровнем технической подготовки и предоставлять им доступ к необходимой информации о продукции и ее характеристиках

2 Выбор инструментальных средств разработки информационной системы

2.1 Введение

В современном мире разработки программного обеспечения, выбор правильных инструментов является критическим фактором успеха любого проекта. Удачное сочетание технологий позволяет повысить эффективность разработки, обеспечить стабильность и масштабируемость создаваемого приложения. В рамках данного проекта, целью которого является создание высокопроизводительного веб-приложения с архитектурой клиент-сервер, был выбран следующий стек:

* **Сервер:** ASP.NET Core Web API
* **ORM:** Entity Framework (EF) Core
* **Клиент:** React
* **База данных:** PostgreSQL
* **Контейнеризация:** Docker

Ниже представлено подробное обоснование выбора каждой из этих технологий, с учетом их функциональных возможностей, совместимости, преимуществ и роли Docker в разработке и развертывании.

**Предварительное замечание о выборе языка:** Для реализации серверной части (ASP.NET Core Web API) был выбран язык C#. Это решение основывается на следующих факторах:

* **Зрелость и надежность платформы .NET:** C# и .NET предоставляют стабильную и зрелую платформу для разработки корпоративных приложений.
* **Производительность:** .NET Core (и его более поздние версии) показывает отличную производительность, особенно при работе с веб-приложениями.
* **Экосистема и инструменты:** .NET имеет богатую экосистему библиотек, фреймворков и инструментов, включая Entity Framework Core, используемый в этом проекте.
* **Сообщество и поддержка:** Microsoft предоставляет обширную документацию, поддержку и активное сообщество разработчиков C#.
* **Типизация:** C# - статически типизированный язык, что способствует более раннему выявлению ошибок на этапе разработки и повышает надежность кода.
* **Ориентация на объектно-ориентированное программирование:** C# предоставляет мощные возможности для объектно-ориентированного программирования, что упрощает разработку сложных приложений.

2.2 Выбор серверного фреймворка — ASP.NET Core Web API

ASP.NET Core — это современный, кроссплатформенный фреймворк от Microsoft для разработки веб-приложений и API.

**Рассмотренные альтернативы:**

* **Node.js (с Express.js):** Популярный фреймворк на JavaScript, часто используемый для разработки API.
* **PHP (с Laravel/Symfony):** Широко распространенный язык и фреймворки для веб-разработки.

**Обоснование выбора ASP.NET Core Web API:**

* **Высокая производительность:** ASP.NET Core демонстрирует превосходную производительность по сравнению с Node.js и PHP, что критично для быстрой отдачи серверных ответов и масштабируемости системы. Тесты производительности (например, TechEmpower Benchmarks) показывают, что ASP.NET Core часто превосходит конкурентов.
* **Кроссплатформенность:** Возможность запуска на различных операционных системах (Windows, Linux, macOS) обеспечивает гибкость развертывания приложений и упрощает разработку в различных средах.
* **Современная архитектура:** ASP.NET Core поддерживает внедрение зависимостей, инверсию управления, а также модульность компонентов, что упрощает сопровождение и расширение кода, делая его более читаемым и поддерживаемым.
* **Широкое сообщество и поддержка:** Большое сообщество разработчиков и официальная поддержка от Microsoft обеспечивают регулярные обновления, исправления ошибок и наличие обширной документации и обучающих материалов.
* **Встроенная поддержка:** ASP.NET Core имеет встроенную поддержку для работы с различными форматами данных (JSON, XML) и протоколами (HTTP, HTTPS), упрощая создание API.
* **Минимализм:** ASP.NET Core предоставляет минимальный набор зависимостей, что позволяет значительно уменьшить размер приложения и упростить развертывание.
* **Интеграция с EF Core:** Отличная интеграция с выбранным ORM Entity Framework Core (EF Core) обеспечивает удобство работы с базами данных.

**Сравнительный анализ (ASP.NET Core vs. Node.js/PHP):**

Таблица 1 — Сравнительный анализ серверных фреймворков

| **Характеристика** | **ASP.NET Core** | **Node.js (Express.js)** | **PHP (Laravel/Symfony)** |
| --- | --- | --- | --- |
| Язык | C# | JavaScript/TypeScript | PHP |
| Производительность | Высокая | Высокая (при оптимизации) | Средняя (зависит от версии PHP и фреймворка) |
| Кроссплатформенность | Да | Да | Да |
| Типизация | Статическая | Динамическая | Динамическая |
| Сложность разработки | Может быть сложной для новичков в C# | Проста в освоении для JavaScript-разработчиков | Проста в освоении (особенно с использованием фреймворков) |
| Экосистема | Богатая (NuGet) | Богатая (npm) | Очень богатая (Composer) |
| Поддержка Microsoft | Да | Нет | Нет |
| ORM интеграция | Отличная (EF Core) | Sequelize, Mongoose, TypeORM и другие | Doctrine, Eloquent (Laravel) и другие |
| Масштабируемость | Высокая | Высокая | Средняя (зависит от архитектуры) |

**Вывод:** ASP.NET Core Web API был выбран из-за его высокой производительности, кроссплатформенности, надежности и интеграции с экосистемой .NET, особенно с EF Core.

2.3 Обоснование выбора ORM — Entity Framework (EF) Core

Для взаимодействия с базой данных в проекте выбран ORM-инструмент Entity Framework Core.

**Рассмотренные альтернативы:**

* **Dapper:** Легковесный ORM, обеспечивающий большую гибкость и производительность.
* **ADO.NET:** Низкоуровневый доступ к базе данных, обеспечивающий максимальный контроль.

**Обоснование выбора EF Core:**

* **Абстракция работы с данными:** EF Core позволяет работать с данными на уровне объектно-ориентированного программирования, что упрощает разработку, снижает вероятность ошибок, связанных с написанием SQL-запросов вручную, и делает код более читаемым.
* **Поддержка миграций:** Механизм миграций упрощает управление изменениями структуры базы данных, что важно при итеративной разработке, позволяя легко обновлять базу данных при изменениях в модели данных.
* **Высокая совместимость с ASP.NET Core:** EF Core тесно интегрирован с ASP.NET Core, что облегчает настройку и использование совместно с Web API, обеспечивая согласованность и удобство разработки.
* **Оптимизация запросов:** EF Core обладает средствами для оптимизации запросов и управления производительностью, позволяя создавать эффективные обращения к базе данных, снижая нагрузку на сервер базы данных.
* **Автоматическое создание SQL-запросов:** EF Core автоматически генерирует SQL-запросы на основе моделей, что упрощает работу с данными.
* **Улучшенная производительность:** EF Core регулярно обновляется и оптимизируется, что обеспечивает хорошую производительность при работе с базами данных.

**Сравнительный анализ (EF Core vs. Dapper/ADO.NET):**

Таблица 2 — Сравнительный ORM-инструментов

| **Характеристика** | **Entity Framework Core** | **Dapper** | **ADO.NET** |
| --- | --- | --- | --- |
| Уровень абстракции | Высокий (объектно-ориентированный) | Низкий (близкий к SQL) | Низкий (прямой доступ к базе данных) |
| Простота разработки | Упрощает разработку, снижает ручной ввод SQL | Требует написания SQL-запросов вручную | Требует написания SQL-запросов и ручного управления соединениями |
| Производительность | Средняя (зависит отвая обучения | Высокая | Более пологая |
| Читаемость кода | Высокая (для объектно-ориентированной логики) | Может быть низкой (зависит от сложности SQL) | Низкая |

**Вывод:** EF Core был выбран из-за удобства работы с данными на объектном уровне, поддержки миграций и тесной интеграции с ASP.NET Core. Dapper - отличный выбор для сценариев, где важна максимальная производительность и контроль над SQL-запросами, однако EF Core обеспечивает более быструю разработку и упрощенное управление структурой базы данных.

2.4 Выбор клиентского фреймворка — React

Для реализации клиентской части выбран JavaScript-фреймворк React.

**Рассмотренные альтернативы:**

* **Angular:** Мощный и полнофункциональный фреймворк от Google.
* **Vue.js:** Простой и гибкий фреймворк с хорошей производительностью.

**Обоснование выбора React:**

* **Компонентный подход:** React построен на компонентах, что обеспечивает переиспользуемость кода, упрощает поддержку и расширение интерфейса, а также способствует модульности и организации кода.
* **Высокая производительность за счет виртуального DOM:** Механизм виртуального DOM помогает минимизировать обновления реального DOM, что повышает отзывчивость пользовательского интерфейса и обеспечивает плавную работу приложения.
* **Богатая экосистема и большое сообщество:** Большое количество готовых библиотек и инструментов (например, для работы с состояниями, маршрутизации, обработки данных) облегчает внедрение дополнительных функций и ускоряет разработку.
* **Гибкость и возможность интеграции:** React легко интегрируется с различными серверными технологиями и API, включая RESTful-сервисы, предоставляемые ASP.NET Core Web API.
* **Комьюнити и популярность:** React широко используется в индустрии, что облегчает поиск разработчиков и поддержку.
* **React-Router-Dom:** Для навигации внутри приложения будет использоваться react-router-dom, что позволит создавать одностраничные приложения (SPA) с множеством страниц и маршрутов, обеспечивая удобство навигации.

**Сравнительный анализ (React vs. Angular/Vue.js):**

Таблица 3 — Сравнительный анализ клиентских фреймворков

| **Характеристика** | **React** | **Angular** | **Vue.js** |
| --- | --- | --- | --- |
| Архитектура | Компонентно-ориентированный (с использованием JSX) | Компонентно-ориентированный (TypeScript, MVC/MVVM) | Компонентно-ориентированный (с использованием шаблонов или JSX) |
| Язык | JavaScript/TypeScript | TypeScript | JavaScript/TypeScript |
| Кривая обучения | Более пологая (основы легко освоить) | Более крутая (большой объем для изучения) | Более пологая (проще, чем Angular) |
| Размер приложения | Может быть минимальным (при грамотной организации) | Большой (за счет функциональности) | Меньше, чем Angular |
| Производительность | Высокая (виртуальный DOM) | Высокая (оптимизация изменений DOM) | Высокая (виртуальный DOM) |
| Экосистема | Богатая (широкий выбор библиотек для управления состоянием, маршрутизации и т.д.) | Богатая (развитая экосистема, поддержка Google) | Хорошая (широкий выбор библиотек, активное сообщество) |
| Управление состоянием | Redux, Context API, Zustand и другие | NgRx, NgRX/Store (похоже на Redux), Services | Vuex, Pinia и другие |
| Шаблонизация | JSX (JavaScript XML) | HTML-шаблоны с директивами | HTML-шаблоны, JSX, или Render-функции |
| Связывание данных | Одностороннее | Двустороннее | Двустороннее (по умолчанию) и одностороннее (при необходимости) |
| Масштабируемость | Высокая (модульность, компонентный подход) | Высокая (структурированность, поддержка больших проектов) | Высокая (компонентный подход, подходит для проектов разного масштаба) |

**Вывод:** React был выбран из-за его простоты изучения, гибкости, обширной экосистемы и высокой производительности. Компонентный подход React позволяет эффективно разрабатывать пользовательские интерфейсы, а также обеспечивает переиспользуемость кода, что является важным аспектом для любого проекта.

2.5 Выбор системы управления базами данных — PostgreSQL

В качестве СУБД была выбрана PostgreSQL — объектно-реляционная система управления базами данных с открытым исходным кодом.

**Рассмотренные альтернативы:**

* **MySQL:** Популярная СУБД с открытым исходным кодом.
* **MS SQL Server:** Коммерческая СУБД от Microsoft.

**Обоснование выбора PostgreSQL:**

* **Надежность и устойчивость:** PostgreSQL признана одной из самых стабильных и надежных СУБД с поддержкой транзакций, ACID-свойств и механизма сохранения данных.
* **Соответствие стандартам SQL:** Поддержка широкого спектра стандартных SQL-конструкций облегчает разработку и миграцию.
* **Расширяемость и поддержка сложных типов данных:** PostgreSQL поддерживает JSONB, массивы, географические типы (PostGIS) и другие расширения, что делает ее подходящей для широкого спектра задач.
* **Высокая производительность:** Оптимизирована под высоконагруженные приложения, обеспечивая эффективную работу с большими объемами данных.
* **Активное сообщество и бесплатная лицензия:** Открытый исходный код снижает затраты на лицензирование и способствует быстрому развитию и поддержке.
* **Отсутствие лицензионных ограничений:** PostgreSQL позволяет использовать базу данных без каких-либо ограничений, что важно для масштабируемых проектов.
* **Поддержка современных функций:** PostgreSQL постоянно обновляется, добавляя новые функции и улучшения, такие как поддержка window functions и оптимизация запросов.

**Сравнительный анализ (PostgreSQL vs. MySQL/MS SQL Server):**

Таблица 4 — Сравнительный анализ СУБД

| **Характеристика** | **PostgreSQL** | **MySQL** | **MS SQL Server** |
| --- | --- | --- | --- |
| Лицензия | Открытый исходный код (BSD) | Открытый исходный | Средняя |
| Надежность | Высокая (транзакции, ACID, WAL) | Высокая (но зависит от конфигурации) | Высокая (транзакции, ACID) |
| Производительность | Высокая (оптимизация запросов, многопоточность) | Высокая (оптимизация для чтения, масштабируемость) | Высокая (оптимизация, инструменты мониторинга) |
| Сложные типы данных | Поддержка (JSONB, массивы, PostGIS) | Ограниченная | Поддержка (XML, пространственные типы) |
| Поддержка Windows | Отличная | Отличная | Отличная |
| Функциональность | Богатая (поддержка оконных функций, рекурсивных запросов) | Хорошая | Богатая (аналитические функции, интеграция с .NET) |

**Вывод:** PostgreSQL была выбрана из-за ее надежности, соответствия стандартам SQL, высокой производительности и поддержки сложных типов данных. Бесплатная лицензия и открытый исходный код также являются важными преимуществами.

2.6 Использование Docker для разработки и развертывания

Для контейнеризации приложения и упрощения процесса развертывания выбран Docker. Docker позволяет упаковать приложение и его зависимости в контейнеры, обеспечивая единообразную среду выполнения, независимо от операционной системы. Это упрощает развертывание, масштабирование и управление приложением.

**Преимущества использования Docker:**

* **Изоляция:** Контейнеры изолированы друг от друга и от хост-системы, что предотвращает конфликты зависимостей и обеспечивает стабильность работы.
* **Переносимость:** Контейнеры Docker легко переносятся между различными средами разработки, тестирования и production, что упрощает процесс развертывания.
* **Масштабируемость:** Docker позволяет легко масштабировать приложение путем создания нескольких контейнеров.
* **Управление зависимостями:** Docker упрощает управление зависимостями приложения, позволяя упаковывать все необходимые библиотеки и инструменты в контейнер.
* **Автоматизация:** Docker автоматизирует процесс сборки, тестирования и развертывания, что сокращает время разработки и улучшает качество.
* **Воспроизводимость:** Контейнеры Docker обеспечивают воспроизводимость окружения, что упрощает отладку и тестирование.

В рамках проекта Docker будет использоваться для контейнеризации:

* **ASP.NET Core Web API:** Для обеспечения стабильности и удобства развертывания серверной части.
* **PostgreSQL:** Для обеспечения согласованной среды разработки и развертывания базы данных.
* **React приложение (опционально):** Для удобства разработки и развертывания клиентской части (использование Docker для frontend зависит от конкретной архитектуры и предпочтений разработчика).

**Сравнительный анализ (Docker vs. другие методы развертывания):**

Таблица 5 — Сравнительный анализ методов развертывания

| **Характеристика** | **Docker** | **Виртуальные машины (VM)** | **Традиционное развертывание (на сервере)** |
| --- | --- | --- | --- |
| Изоляция | Высокая (контейнеры) | Высокая (VM) | Низкая (зависимости могут конфликтовать) |
| Ресурсы | Эффективное использование (меньше ресурсов, чем VM) | Высокое потребление ресурсов (каждая VM - отдельная ОС) | Зависит от конфигурации сервера и приложений |
| Переносимость | Высокая (Docker images) | Высокая (VM images) | Низкая (зависит от ОС и конфигурации) |
| Масштабируемость | Легкая (быстрое создание и удаление контейнеров) | Более сложная (запуск новых VM) | Сложная (требует настройки и развертывания приложений) |
| Управление | Упрощенное (Docker Compose, Kubernetes) | Сложное (управление несколькими VM) | Сложное (требует ручной настройки серверов) |
| Время развертывания | Быстрое (секунды/минуты) | Долгое (минуты/часы) | Зависит от сложности развертывания |
| Автоматизация | Высокая (CI/CD, Dockerfiles) | Высокая (но требует дополнительных инструментов) | Средняя (требует скриптов и инструментов автоматизации) |
| Размер | Небольшой | Большой (ОС, приложения, зависимости) | Зависит от размера приложений и зависимостей |

**Вывод:** Использование Docker позволяет обеспечить:

* **Согласованную среду:** Упрощает настройку и поддержание среды разработки, делая ее одинаковой для всех разработчиков.
* **Быстрое развертывание:** Ускоряет процесс развертывания на различных серверах.
* **Масштабируемость:** Упрощает масштабирование приложения при увеличении нагрузки. Docker является отличным выбором для современных веб-приложений, обеспечивая гибкость, эффективность и упрощенное управление.

2.7 Вывод по главе

Таким образом, выбранный стек инструментальных средств (ASP.NET Core Web API, Entity Framework Core, React, PostgreSQL, Docker) обеспечивает высокую производительность, надежность и удобство разработки на всех уровнях программного продукта.

3 Проектирование и разработка интерактивного конфигуратора

3.1 Анализ требований к конфигуратору

Данный раздел содержит анализ требований к конфигуратору, разделенных на функциональные и нефункциональные требования.

3.1.1 Функциональные требования

Функциональные требования описывают функциональность, которую должен предоставлять конфигуратор для пользователей.

1. **Управление каталогом продукции:** Обеспечение возможностей добавления, редактирования и удаления продукции из каталога, а также загрузки изображений и технических характеристик.

* Добавление продукции: Реализация возможности добавления новой продукции в каталог с указанием следующих параметров: название, артикул/код, описание (краткое и подробное), категория, цена, производитель. Должна быть обеспечена возможность сохранения данных о новом продукте в базе данных.
* Редактирование продукции: Реализация возможности изменения информации о существующей продукции в каталоге. Предоставление возможности редактирования всех полей, доступных при добавлении продукта. Изменения должны быть сохранены в базе данных.
* Удаление продукции: Реализация возможности удаления продукции из каталога с обязательным подтверждением действия. Удаление должно удалять данные продукта из базы данных.
* Загрузка изображений: Предоставление возможности прикрепления изображений к продукции, с поддержкой распространенных графических форматов (например, JPG, PNG, GIF) и возможностью оптимизации изображений для отображения на веб-страницах. Предусмотреть хранение изображений (на сервере).
* Дополнительные возможности:  Предусмотреть категоризацию продукции, поиск по каталогу (по названию, артикулу, описанию и другим параметрам), фильтрацию по каталогу (по категориям, производителям и другим параметрам).

1. **Конфигуратор оборудования:** Преющие списки, переключатели, ползунки, текстовые поля) и визуального отображения выбранных значений.

* Ввод данных пользователем: Реализация текстовых полей ввода и выпадающих списков для ввода параметров оборудования.
* Подбор оборудования: Реализация алгоритма, который на основе введенных данных подбирает подходящие модели оборудования из каталога.
* Отображение результата: Отображение карточек подходящих продуктов с информацией (название, изображение, краткое описание, цена).

1. **ИИ-Консультант:** Реализация чат-бота, способного вести диалог с пользователем в реальном времени. Возможность подбора оборудования через диалог с чат-ботом.

3.1.2 Нефункциональные требования

Нефункциональные требования определяют характеристики системы, которые не относятся непосредственно к функциональности, но влияют на ее качество.

**Производительность:** Конфигуратор должен обеспечивать быструю загрузку данных и мгновенный отклик на действия пользователя. Время загрузки каталога продукции и данных о конфигурации должно быть минимальным. Необходимо использовать оптимизированные запросы к базе данных, кэширование данных, оптимизацию изображений и технологию ленивой загрузки. Система должна обеспечивать минимальное время отклика на действия пользователя и быть способной обрабатывать большое количество данных и пользователей без снижения производительности.

**Безопасность:** Конфигуратор должен обеспечивать защиту от несанкционированного доступа и взлома. Необходимо реализовать аутентификацию и авторизацию пользователей, защиту от SQL-инъекций, межсайтового скриптинга (XSS), CSRF, шифрование данных и регулярные обновления безопасности. Должна быть предусмотрена система резервного копирования и восстановления данных.

**Удобство использования:** Конфигуратор должен быть интуитивно понятным и простым в использовании даже для пользователей с небольшим опытом работы с компьютером. Интерфейс должен быть понятным, логичным и адаптированным для различных устройств. Необходимо предоставлять обратную связь пользователям и обеспечить наличие документации и помощи.

3.1.3 Технические требования

* Технологии: ASP.NET Core Web API, EF, React.
* База данных: Postgresql.
* API: GigaChat.
* Развертывание: Docker.

3.2 Проектирование архитектуры системы

Архитектура системы конфигуратора состоит из трех основных компонентов: Frontend, Backend и Database.

**Frontend (React.js)** отвечает за предоставление пользовательского интерфейса для взаимодействия с конфигуратором. Он разработан с использованием React.js, JavaScript (ES6+), HTML и CSS (или CSS-in-JS библиотеки). Для управления состоянием приложения может использоваться Redux или Context API. Основные функции включают отображение каталога продукции и интерфейса конфигурации, обработку пользовательского ввода, взаимодействие с Backend API, отображение результатов конфигурации и обеспечение удобного интерфейса. При проектировании учитывается компонентная архитектура React.js, оптимизация производительности, адаптивный дизайн и обработка ошибок.

**Backend (ASP.NET Core Web API)** предоставляет API для доступа к данным и выполнения бизнес-логики. Он разработан с использованием ASP.NET Core, C#, .NET Runtime и Entity Framework Core для работы с базой данных. Основные функции включают обработку HTTP-запросов, авторизацию и аутентификацию пользователей, взаимодействие с базой данных, реализацию бизнес-логики конфигуратора, формирование ответов в формате JSON и предоставление API для управления каталогом продукции. При проектировании учитывается RESTful API, использование Dependency Injection, реализация обработки ошибок и, использование индексов, резервное копирование, безопасность базы данных и выбор типа данных для каждого поля.

**Схема взаимодействия компонентов:**

1. Пользователь взаимодействует с Frontend (React.js).
2. Frontend отправляет HTTP-запросы к Backend API (ASP.NET Core Web API).
3. Backend обрабатывает запросы, взаимодействует с базой данных (PostgreSQL).
4. Backend возвращает ответы в формате JSON Frontend.
5. Frontend отображает данные пользователю.

**Преимущества данной архитектуры:**

* Разделение ответственности: Каждый компонент отвечает за свою конкретную задачу, упрощая разработку, тестирование и поддержку.
* Масштабируемость: Каждый компонент может быть масштабирован независимо.
* Гибкость: Можно легко заменить один компонент другим.
* Поддержка современных технологий: Использование React.js, ASP.NET Core и PostgreSQL обеспечивает современные подходы к разработке.
* Независимое развертывание: Каждый компонент можно развертывать независимо.

Архитектура системы представлена на рискнке:



Рис. 1. Архитектура системы

3.3 Проектирование базы данных

3.3.1 Проектирование инфологической модели БД

Проектирование инфологической модели является основной задачей при создании БД. Цель инфологической модели – обеспечение наиболее естественных для человека способов сбора и представления той или иной информации, которую предполагается хранить в создаваемой базе. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком (последний не может быть использован в чистом виде из-за сложности компьютерной обработки текстов и неоднозначности любого естественного языка). Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства.

Поэтому необходимо четко поставить цель данной БД, а также установить, какую информацию получает пользователь в результате работы с программой.

Для данной ИС были выделенs следующие сущности:

1. Устройство
2. Тип устройства
3. Тип помещения
4. Пользователь

В ходе проектирования была разработана следующая концептуальная модель базы данных:

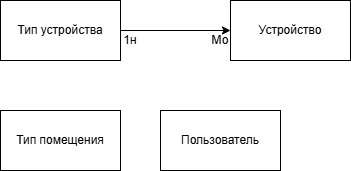


Рис. 2. Концептуальная ER-Диаграмма

3.3.2 Проектирование даталогической модели БД

Проектирование даталогической модели - важный этап в проектировании базы данных. На этом этапе важно правильно выделить сущности и описать их атрибуты. Ошибка на этом этапе может обернуться разработчику значительными потерями времени и сил в дальнейшем.

Датологическая модель - модель, описывающая логику организации данных. Датологическое проектирование заключается в проектировании логической структуры БД. Таким образом, главное отличие даталогической модели от инфологической состоит в том, что инфологическая модель хранит в себе всю информацию о предметной области, необходимую и достаточную для проектирования базы данных, но она не привязана к определенной СУБД.

Даталогическое проектирование сводится к следующим этапам:

1. Определение таблиц;

2. Определение полей таблиц;

3. Определение типов данных в соответствии с выбранной СУБД;

4. Определение длины каждого поля таблиц;

5. Определение обязательности каждого поля;

6. Определение индексации каждого поля.

Далее будут представлены сформированные таблицы:

Таблица 1 - Устройство, ключевое поле «ID»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничения |
| ID | Счетчик | Номер по порядку уникален |
| Название | Текстовый | 255 значений, обязательное поле: да |
| Описание | Текстовый | 255 значений, обязательное поле: да |
| Путь к изображению | Текстовый | 255 значений, обязательное поле: нет |
| Мощность | Числовой | Длинное целое, обязательное поле: да |
| Уровень шума | Числовой | Длинное целое, обязательное поле: да |
| Максимальный поток воздуха | Числовой | Длинное целое, обязательное поле: да |
| Цена | Числовой | Длинное целое, обязательное поле: да |
| ID Тип устройства | Числовой | Длинное целое, обязательное поле: да |

Таблица 2 – Тип устройства, ключевое поле «ID»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничения |
| ID | Счетчик | Номер по порядку уникален |
| Название | Текстовый | 255 значений, обязательное поле: да |
| Описание | Текстовый | 255 значений, обязательное поле: да |

Таблица 3 – Тип помещения, ключевое поле «ID»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничения |
| ID | Счетчик | Номер по порядку уникален |
| Коэффициент площади | Числовой | Дробное, обязательное поле: да |
| Коэффициент людей | Числовой | Дробное, обязательное поле: да |

Таблица 4 – Пользователь, ключевое поле «ID»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Ограничения |
| ID | Счетчик | Номер по порядку уникален |
| Email | Текстовый | 255 значений, обязательное поле: да |
| Пароль | Текстовый | 255 значений, обязательное поле: да |
| Роль | Текстовый | 255 значений, обязательное поле: да |
| Дата создания | Дата/время | Обязательное поле: да |

Заключение

В ходе прохождения технологической (проектно-технологической) практики в Акционерном обществе «ТИОН Умный микроклимат» была проведена работа, направленная на проектирование интерактивного конфигуратора для продукции компании. В рамках практики был выполнен всесторонний анализ деятельности компании, включая изучение бизнес-процессов, связанных с конфигурацией продукции, и анализ целевой аудитории.

На основе проведенного анализа и с учетом поставленных задач было выбрано оптимальное технологическое решение для разработки конфигуратора. Выбор был сделан в пользу следующих технологий: ASP.NET Core Web API для создания серверной части, Entity Framework Core для упрощения взаимодействия с базой данных, React для разработки клиентской части, PostgreSQL в качестве СУБД, и Docker для контейнеризации и развертывания приложения. Данный выбор был обусловлен высокой производительностью, масштабируемостью и удобством разработки, которые позволят реализовать эффективный и удобный конфигуратор. Разработанный конфигуратор призван улучшить взаимодействие с клиентами, автоматизировать процесс подбора оборудования и повысить эффективность работы менеджеров по продажам, что в конечном итоге оптимизирует бизнес-процессы компании. Результаты практики подтверждают возможность успешного внедрения предложенного решения для оптимизации бизнес-процессов компании.

Список литературы

1. Агальцов, В. П. Базы данных: в 2 книгах. Книга 2. Распределенные и удаленные базы данных: учебник / В.П. Агальцов. — Москва: ФОРУМ : ИНФРА-М, 2021. — 271 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-8199-0713-9. - Текст: электронный. - URL: https://znanium.com/catalog/product/1514118 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

2. Жуков, Р. А. Базы данных: учебно-методическое пособие по дисциплине «Базы данных» для направления подготовки 38.03.05 «Бизнес-информатика» (бакалавриат) / Р. А. Жуков. - Москва; Берлин: Директ-Медиа, 2019. - 176 с. - ISBN 978-5-4499-0225-2. - Текст: электронный. - URL: https://znanium.com/catalog/product/1874923 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

3. Шустова, Л. И. Базы данных: учебник / Л.И. Шустова, О.В. Тараканов. — Москва: ИНФРА-М, 2021. — 304 с. + Доп. материалы [Электронный ресурс]. — (Высшее образование: Бакалавриат). — DOI 10.12737/11549. - ISBN 978-5-16-010485-0. - Текст: электронный. - URL: https://znanium.com/catalog/product/ 1362122 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

4. Белов, В. В. Алгоритмы и структуры данных : учебник / В. В. Белов, В. И. Чистякова. - Москва: КУРС: ИНФРА-М, 2020. - 240 с. - (Бакалавриат). - ISBN 978-5-906818-25-6. - Текст: электронный. - URL: https://znanium.com/catalog/ product/1057212 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

5. Хорев, П. Б. Объектно-ориентированное программирование с примерами на С# : учебное пособие / П.Б. Хорев. — Москва : ФОРУМ : ИНФРА-М, 2020. — 200 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-00091-680-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1069921 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

6. Кокоса, К. Управление памятью в .NET для профессионалов : практическое руководство / К. Кокоса. - Москва : ДМК Пресс, 2020. - 800 с. - ISBN 978-5-97060-800-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1210679 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

7. Новиков, Б. А. Основы технологий баз данных : учебное пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева ; под ред. Е. В. Рогова. — 2-е изд. — Москва : ДМК Пресс, 2020. - 582 с. - ISBN 978-5-97060-841-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/1210665 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

8. Сьоре, Э. Проектирование и реализация систем управления базами данных : учебное пособие / Эдвард Сьоре ; пер. с анг. А. Н. Киселева ; научн. ред. Е. В. Рогов. - Москва : ДМК Пресс, 2021. - 466 с. - ISBN 978-5-97060-488-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1225360 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

9. Федотов, И. Е. Параллельное программирование. Модели и приемы : практическое пособие / И. Е. Федотов. - Москва : СОЛОН-Пресс, 2020. - 390 с. - (Серия «Библиотека профессионала»). - ISBN 978-5-91359-222-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1858781 (дата обращения: 04.05.2022). – Режим доступа: по подписке.

10. Торстейнсон, П. Криптография и безопасность в технологии .NET / П. Торстейнсон, Г. А. Ганеш. - 4-е изд. - Москва : Лаборатория знаний, 2020. - 482 с. - (Программисту). - ISBN 978-5-00101-700-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1201346 (дата обращения: 04.05.2022). – Режим доступа: по подписке.