Министерство науки и высшего образования Республики Казахстан
НАО «Казахский национальный университет имени аль-Фараби»

УДК 004.4

УТВЕРЖДАЮ
Член Правления - Проректор
по научно-инновационной деятельности
_____ Ж.Н. Айтжанова
«_____» _____ 2023 г.

ОТЧЕТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Создание прикладных программных продуктов для анализа разработанных технологий на основе концепции Digital Twin (цифровой двойник)

по теме:
Проектирование архитектуры программного продукта для создания новых технологических решений комплексной переработки сложного металлургического сырья

в рамках программы:
Создание новых технологических решений комплексной переработки сложного металлургического сырья в соответствии с концепцией «Индустрии 4.0» и Digital Twin
(промежуточный, BR21882140)

Научный руководитель: Т.С. Иманкулов

Алматы 2023

СПИСОК ИСПОЛНИТЕЛЕЙ

Научный руководитель:			
Ведущий научный сотрудник, PhD	_____	Т.С. Иманкулов	(введение, заключение, разделы 1 - 4)
Ответственный исполнитель:			
Научный сотрудник	_____	Е.С. Нурахов	(введение, заключение, разделы 1 - 4)
Исполнители:			
Ведущий научный сотрудник	_____	Б.С. Дарибаев	(разделы 1-4)
Старший научный сотрудник	_____	О.Н. Тұрар	(раздел 1, 3)
Научный сотрудник	_____	Е.Ғ. Кенжебек	(раздел 1)
Научный сотрудник	_____	Н.М. Қасымбек	(раздел 3)
Научный сотрудник	_____	М.Б. Мұстафин	(раздел 2)
Младший научный сотрудник	_____	Б.С. Амангелды	(раздел 2)
Младший научный сотрудник	_____	Н.М.Тасмурзаев	(раздел 1)
Младший научный сотрудник	_____	А.А. Муханбет	(раздел 2)
Младший научный сотрудник	_____	Э.Б. Толеубекова	(раздел 2)
Младший научный сотрудник	_____	А. Еркос	(раздел 4)
Младший научный сотрудник	_____	Ш.Ж. Шинасылов	(раздел 4)
Инженер-программист	_____	Н. Азатбекұлы	(раздел 3)
Инженер-программист	_____	С.Ж. Айбагаров	(раздел 2)

СОДЕРЖАНИЕ

РЕФЕРАТ

Отчет 24 с., 2 рис., 17 источников.

Ключевые слова: МЕТАЛЛУРГИЧЕСКАЯ ПРОМЫШЛЕННОСТЬ, КОМПЛЕКСНАЯ ПЕРЕРАБОТКА СЫРЬЯ, ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ, АРХИТЕКТУРНАЯ КОНЦЕПЦИЯ, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, MYSQL, SQLITE, ASP.NET CORE, ОПТИМИЗАЦИЯ ПРОИЗВОДСТВА, ПРОТОТИПИРОВАНИЕ, ТЕСТИРОВАНИЕ, ОТЛАДКА, МОНИТОРИНГ И ОПТИМИЗАЦИЯ ПРОИЗВОДИТЕЛЬНОСТИ, БЕЗОПАСНОСТЬ, МОДЕЛИРОВАНИЕ И АНАЛИЗ, ЭКСПЕРТНЫЕ ОЦЕНКИ.

Объекты исследования: проектирование архитектуры программного продукта.

Цель работы: Целью исследования является разработка архитектуры программного продукта, которая обеспечит эффективное решение задач комплексной переработки сложного металлургического сырья.

Методы исследований: анализ требований, исследование литературы, проектирование архитектуры, сравнительный анализ.

Результаты работы и их новизна: разработана архитектурная концепция программного продукта, включая описание ключевых компонентов и их взаимодействие. Это обеспечило четкое представление о структуре системы. Выбраны и обоснованы технологии, используемые в разработке, включая языки программирования, фреймворки, базы данных и инструменты интеграции. Это позволило оптимизировать производительность и эффективность продукта. Подготовлена подробная архитектурная документация, включая диаграммы классов, последовательности и компонентов, а также описания компонентов и интерфейсов.

Программный продукт предоставляет возможность оптимизировать производственные процессы в металлургической промышленности, что способствует снижению затрат и повышению качества продукции. Данная система предлагает комплексное решение для металлургической и горнодобывающей промышленности, уделяя особое внимание эффективному анализу данных и расчетам. Он поддерживает разнообразные вычислительные методы и алгоритмы для обработки сложных отраслевых данных. Благодаря интуитивно понятному и удобному интерфейсу он позволяет легко визуализировать результаты расчетов и легко интегрируется с внешними ресурсами. Система разработана для высокоскоростных вычислений и масштабируемой обработки данных, обеспечивая надежную производительность. Подчеркивая безопасность, он включает в себя строгие меры аутентификации пользователей и защиты данных. Придерживаясь высоких стандартов кодирования, система проходит тщательное тестирование на качество и безопасность. Разработанный для горизонтального масштабирования, он обеспечивает сбалансированное распределение нагрузки и облегчает интеграцию с различными исследовательскими и вычислительными инструментами, что знаменует собой значительный прогресс в технологиях металлургии и переработки полезных ископаемых.

ВВЕДЕНИЕ

Одним из современных и технологических инструментов для решения комплексной переработки сложного металлургического сырья является десктопное приложение, которое позволяет выполнять сложные математические расчеты с помощью простого и понятного User-интерфейса (пользовательского интерфейса). Этапы его разработки следует разделить на две части: планирование и проектирование будущего функционала разрабатываемого десктопного приложения; его непосредственная разработка с использованием передовых технологии и фреймворков.

Десктопное приложение будет выполнять важные функции и будет предоставлять понятный интерфейс для ввода, редактирования и вывода необходимых данных, а также генерировать анализ результатов. Это позволит многократно упростить вычисления процесса переработки металлургического сырья. Более подробное описание планируемого функционала десктопного приложения, что касается ввода данных:

- Ввод параметров сырья. В приложении будет возможность ввода информации о характеристике металлургического сырья, такие как его состав, физические свойства, размер частиц и другие параметры, которые могут варьироваться в зависимости от вида сырья.
- Настройка технологических параметров. Приложение будет иметь не только возможность ввода, но и регулировки различных параметров, таких как температура, давление, скорость процесса и другие важные факторы, определяющие характер переработки.
- Сохранение и загрузка настроек. Пользователь должен иметь возможность сохранять настроенные параметры для последующего использования или загрузки ранее сохраненных конфигураций. Это упрощает повторное использование определенных настроек и обмен конфигурациями между пользователями.
- Валидация данных. Приложение будет предоставлять механизмы для обратной связи с пользователем, такие как, предупреждения об ошибках ввода данных и проверку на валидность введенных параметров для предотвращения некорректных результатов.

В контексте создания десктопных приложений для комплексной переработки металлургического сырья, инструменты визуализации данных играют ключевую роль в обеспечении пользовательского опыта и эффективности анализа. Немаловажный элемент любого десктопного приложения – это его визуализации, позволяющие эффективно анализировать данные, введенные пользователем, параметры и т.д. В контексте создания десктопных приложений для комплексной переработки металлургического сырья это играет гораздо большую роль. Для текущей задачи уместным будет использование следующих видов визуализации:

- Графики. Возможность построения синхронизированных графиков, отображающих изменения параметров во времени. Например, график зависимости температуры от времени или изменения концентрации металлов в процессе переработки.
- Диаграммы. Использование круговых или столбчатых диаграмм для иллюстрации процентного содержания различных компонентов в сырье или конечном продукте.
- Тепловые карты. Визуализация данных в виде тепловых карт может помочь выделить области с наибольшим воздействием параметров. Например, можно использовать тепловые карты для отслеживания зон с наивысшей температурой или давлением внутри реактора.
- 3D-Модели. Трехмерное моделирование может визуализировать пространственное распределение параметров в сложных процессах переработки.
- Интерактивные элементы. Включение элементов управления, таких как бегунки, кнопки, или выпадающие списки, позволяет пользователям манипулировать параметрами в режиме реального времени и сразу видеть результаты.

Самой комплексной и трудоемкой частью в процессе разработки любого приложения, является разработка бэкенда.

Важнейшей частью архитектуры всей системы является ядро сервера, где реализуются основные функции системы, обеспечивающие ее работоспособность и взаимодействие с другими компонентами системы. Данные с пользовательского интерфейса (десктопного приложения) должны передаваться к ядру. Поэтому важным этапом является разработка программного интерфейса (API), обеспечивающего эффективное взаимодействие между десктопным приложением и центральным сервером. Это включает создание API на серверной стороне с использованием протоколов RESTful API. Поскольку через API передаются чувствительные данные, важно обеспечить их безопасность. Использование шифрования (например, протокол HTTPS) обеспечит защищенную передачу данных между десктопным приложением и сервером, предотвращая несанкционированный доступ. Также дополнительно есть возможность применения алгоритмов шифрования RSA или HMAC. Что касается выбора формата передачи данных, использование формата JSON упрощает структурирование и передачу информации между клиентом и сервером, облегчая анализ данных на сервере. Для обеспечения безопасности и управления доступом к серверу будут использоваться механизмы аутентификации. Как правило в его состав входит токены, логины, пароли, а также механизмы проверки подлинности во избежание несанкционированного доступа к данным пользователя программного обеспечения. Важными дополнением в десктопном приложении является система логирования, который поможет отслеживать процесс передачи данных и выявлять возможные проблемы, что может быть полезно и разработчикам, и пользователю. Указанная работа описывает архитектурную концепцию разработки программного продукта, предназначенного для комплексной переработки сложного металлургического сырья. Программное обеспечение разрабатывается для Института металлургии и обогащения, который является крупным научным центром Республики Казахстан и занимается фундаментальными и прикладными исследованиями, технологическими разработками, а также образовательной деятельностью в области металлургии и обогащения.

Целью данного программного продукта является обеспечение возможности проведения исследований и вычислений в области металлургии и обогащения с использованием современных технологий и методов. Программное обеспечение разрабатывается с учетом разнообразных потребностей и требований, предъявляемых к обработке данных, вычислениям и интеграции различных вычислительных методов.

В данном документе подробно описывается архитектурная концепция этого программного продукта, начиная с обзора требований и заканчивая оптимизацией производительности и выявлением узких мест. Описанные в документе решения и стратегии направлены на обеспечение высокой производительности, масштабируемости, безопасности и качества кода.

Настоящее введение служит общим представлением о содержании документа и его целях, а также обосновывает актуальность и важность архитектурной концепции для успешной разработки программного продукта для Института металлургии и обогащения.

1 ОБЗОР ТРЕБОВАНИЙ К ПРОГРАММНОМУ ПРОДУКТУ ДЛЯ КОМПЛЕКСНОЙ ПЕРЕРАБОТКИ СЛОЖНОГО МЕТАЛЛУРГИЧЕСКОГО СЫРЬЯ

В области обогащения и металлургии проводятся научные исследования, связанные с обработкой материалов, которые требуют создания программных продуктов для эффективной обработки и анализа данных в данной сфере.

Использование методов, инструментов и подходов информационных технологий совместно играют важную роль в развитии направления обогащения и металлургии. Они способствуют оптимизации процессов, улучшению производительности и управлению данными. Это позволяет более эффективно осуществлять мониторинг, анализ и оптимизацию производственных операций. В итоге, интеграция этих инструментов в сфере обогащения и металлургии способствует совершенствованию методов и технологий, повышению качества продукции и экономии ресурсов, что способствует дальнейшему развитию этой отрасли.

К примеру, инструмент DataHub подключается к ПЛК или установкам РСУ и считывает информацию о процессах измельчения, разделения, флотации, сгущения или фильтрации. Затем он публикует данные в режиме реального времени на специально разработанных веб-страницах. Системные администраторы имеют возможность настроить дисплеи только для мониторинга или разрешить контрольный контроль [1].

В научной работе [2] с помощью Database Toolbox также обеспечивалась плавная интеграция как с различными используемыми архиваторами процессов — InSQL (Wonderware) и PI (OSIsoft), так и с лабораторной базой данных SQL (Microsoft). Разработанное программное обеспечение не только позволило проводить стандартизированный автоматизированный анализ данных, включающий сбор, проверку, анализ и отчетность данных, но также позволило повторно использовать предварительно обработанные данные для специального ручного анализа данных.

В данной работе [3] спроектированы и введены в эксплуатацию в золотодобывающей, медной, никелевой, магниевой, угольной и горнодобывающей промышленности. в этой работе спроектировал, сконфигурировал, ввел в эксплуатацию и обеспечил поддержку нескольких систем архиватора OSIsoft PI, а также интегрировал архиватор PI с различными внешними системами (базами данных SQL, REST API).

Решения на основе Docker-контейнеров стали переходной точкой для подходов к виртуализации. Фактически, контейнерные решения Docker можно рассматривать как альтернативу традиционным виртуальным машинам. Совместная работа на основе Docker-контейнеров помогает перенести различные программные приложения на облачный уровень [4].

Система [5] в кластере Docker Swarm, которая автоматически развертывает и распределяет сборщики данных и другие задействованные сервисы между различными узлами кластера, где каждый сервис независимо работает над предоставлением высоко доступных коллекций информации конечной точке для использования приложением.

В текущем исследовании [6] оцениваются вычислительные показатели, такие как ЦП, скорость чтения диска, скорость записи на диск, память, сетевой прием и передача, для сравнения веб-приложения в монолитной архитектуре с тем же приложением, использующим микросервисы. Математическая модель, основанная на методе непараметрической регрессии, была применена для проверки результатов этих исследований. Кроме того, в данном проекте могут точно подтвердиться результаты и предоставить количественную техническую интерпретацию. Архитектура микросервисов с контейнерами оказалась более эффективной.

В настоящей работе [8] предлагается архитектура рынка данных как услуги (DaaS), размещенного исключительно в облачной среде. Архитектура включает в себя механизм управления хранилищем, который обеспечивает требования к качеству обслуживания (QoS), компонент мониторинга, который позволяет в реальном времени принимать решения об

используемых ресурсах, и механизм разрешения, который обеспечивает обнаружение и ранжирование семантических данных на основе пользовательских запросов [7]. Целью данного исследования является предложить и оценить подход к миграции монолитной базы данных в мультимодельную многоязычную сохраняемость на основе микросервисной архитектуры.

Целью данной работы [9] является оценка и сопоставление производительности архитектурных шаблонов, наиболее часто используемых для разработки приложений систем (т. е. сервис-ориентированной архитектуры (SOA) и архитектуры микросервисов (MSA)), используя в качестве эталона количественные значения, полученные из различных тесты производительности и их способность адаптироваться к требуемому атрибуту программного обеспечения (т. е. универсальной высокой производительности).

В работе [10] предлагаемое программное обеспечение, написанное на Python, применяется для оценки кинетической базы данных для ГЦК-сплава Co-Cr-Fe-Mn-Ni с высокой энтропией с использованием только данных диффузии индикаторов для четкого разделения термодинамических и кинетических данных. Созданная база данных действительна для всего диапазона составов пятикомпонентного высокоэнтропийного сплава.

Пакет [11] позволяет обучать с использованием различных алгоритмов линейной регрессии с регуляризацией и без нее, байесовской регрессии, выбора признаков и перекрестной проверки. Даннаяа система также предоставляет дополнительные функции для перечисления и сопоставления структур, а также управления и анализа данных. Потенциальные применения иллюстрируются двумя примерами, включая расчет фазовой диаграммы прототипа металлического сплава и анализ химического упорядочения в неорганическом полупроводнике.

В данной работе [12] представлен новый пакет Python PyRK (Python for Reactor Kinetics). PyRK был разработан для моделирования в нулевых измерениях нестационарного, связанного, теплогидравлического и нейтронно-физического поведения, зависящего от времени, в ядерных реакторах.

В данной работе [13] описана важность Python в химии что обеспечивает универсальность, простоту и богатую экосистему библиотек, что делает его предпочтительным выбором для решения химических задач. Он широко используется для кинетических и термодинамических расчетов, а также в квантовой химии и молекулярной механике. Python широко используется для автоматизации лабораторий и разработки программного обеспечения. Анализ данных и визуализация в химии также стали проще благодаря библиотекам, доступным на Python. В будущем ожидается развитие теоретической и вычислительной химии, особенно на пересечении с другими областями, такими как машинное обучение. В обзоре представлены инструменты, разработанные для приложений в кинетической, термодинамической и квантовой химии, приборы для молекулярной механики и лабораторное оборудование.

В данной работе [14] представлено kMCpy, легкий пакет Python с открытым исходным кодом для выполнения kMC-моделирования ионного транспорта в кристаллических твердых телах с использованием данных расчетов DFT. kMCpy и реализованный на нем рабочий процесс предоставляют научному сообществу основу для прогнозирования транспортных свойств любого кристаллического твердого тела с высокой точностью и производительностью. Конструкция kMCpy должна облегчать его использование на большинстве доступных вычислительных платформ, от стандартных ноутбуков до высокопроизводительных.

В работе [15] подходящая база данных была выбрана исходя из наличия химических элементов, минералов и газа. Для моделирования явления скейлинга предложены термодинамическая и кинетическая модели. Термодинамическая модель дала представление о возможных минералах, выпавших в осадок, в то время как кинетическая модель, после

изменения начального уравнения скорости, дала результаты, которые были близки к ожидаемому составу отложений на геотермальной электростанции SsF.

В статье [16] представлены новые структуры для анализа сплавов в контексте баз данных материалов, внедрили ее в пакет pymatgen-analytics-alloys с открытым исходным кодом и создали базу данных сплавов с открытым исходным кодом, которая была включена в MP Web. Здесь представлено несколько тематических исследований того, как эту базу данных можно использовать в контексте исследования и проектирования материалов.

API для доступа и поиска в базе данных определен в коде emmet с открытым исходным кодом. Пользовательский интерфейс на веб MP создан с использованием веб-инфраструктуры Crystal Toolkit с открытым исходным кодом.

Термодинамические и кинетические данные обычно необходимы для количественного моделирования обработки, структуры и свойств материалов. Программные интерфейсы Thermo-Calc, включая TQ, TC-API и TC MATLAB Toolbox, предоставляют прикладным программистам иерархию API-интерфейсов для доступа к термодинамическим и кинетическим данным через ядро Thermo-Calc и DICTRA, наиболее широко используемое программное обеспечение и систему баз данных. для расчета многокомпонентного фазового равновесия и фазовых превращений. С помощью этих программных интерфейсов можно напрямую получать изменения термодинамических и кинетических свойств в реальном времени при изменении местной температуры, давления или состава. В этой статье будут представлены структура и использование интерфейсов программы Thermo-Calc. Будут проиллюстрированы успешные примеры применения [17].

В области разработки программного обеспечения для металлургической обработки данный проект представляет революционную микросервисную архитектуру, которая переопределяет эффективность и функциональность основных прикладных систем. В основе этой инновации лежит микросервис ядра приложения, который имеет уникальные возможности доминировать в архитектуре системы, управляя такими важными функциями, как обработка данных и оптимизация процессов. Это ядро тщательно спроектировано для поддержки бизнес-логики и правил, обеспечивая бесперебойную работу.

Отличительной чертой этого проекта является синергетическая интеграция ядра со специализированными микросервисами, такими как вычислительный микросервис, который использует передовые алгоритмы для точных металлургических расчетов. Такая конструкция не только повышает точность вычислений, но также повышает общую оперативность и эффективность системы. Кроме того, сложный микросервис пользовательского интерфейса, ориентированный на доступность и адаптивный дизайн, значительно улучшает взаимодействие с пользователем, делая сложные процессы более интуитивно понятными и удобными для пользователя.

Этот новый подход к разработке ядра приложения на основе микросервисов в сочетании с особым вниманием к пользовательскому интерфейсу и эффективности вычислений ставит данный проект в авангарде технологических достижений в области программного обеспечения для металлургии. Этот проект означает не только прорыв в разработке приложений, но и новую парадигму подхода, управления и оптимизации металлургической обработки и расчетов в цифровую эпоху.

Метод диффузии. Программное обеспечение принимает исходные составы и структуры материалов, а также заданные коэффициенты диффузии. Пользователи вводят температурные профили и продолжительность времени. Программное обеспечение рассчитывает профили концентрации с течением времени, используя уравнения диффузии (например, законы Фика). Результаты включают визуализацию градиентов концентрации и потенциальных фазовых изменений.

Метод осаждения. Пользователи вводят исходный состав сплавов, термодинамические данные и кинетические параметры. Указаны температурные профили и временные рамки

осадков. Программное обеспечение использует модели кинетики нуклеации и роста для моделирования эволюции осадков. Выходные данные обычно включают распределение размеров, объемную долю осадков и изменения свойств материала.*3.*

Раздел "Обзор требований к программному продукту для комплексной переработки сложного металлургического сырья" представляет собой детальное описание ключевых требований и характеристик, которые должны быть учтены при разработке данного программного продукта. Эти требования определяют основные функциональные, производственные и безопасностные параметры продукта. В данном разделе разрабатываемого проекта обобщаются основные требования к программному продукту:

Изучив эти решения были сформированы (Приложение 1) требования обеспечивают основу для разработки архитектурной концепции, которая будет обеспечивать успешное выполнение задач, поставленных перед программным продуктом для комплексной переработки сложного металлургического сырья.

2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОГРАММНОГО ПРОДУКТА ДЛЯ СОЗДАНИЯ НОВЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ КОМПЛЕКСНОЙ ПЕРЕРАБОТКИ СЛОЖНОГО МЕТАЛЛУРГИЧЕСКОГО СЫРЬЯ

Микросервисная архитектура в веб-приложении для расчета сплавов представляет собой передовую парадигму разработки программного обеспечения, характеризующуюся ее декомпозицией на набор слабосвязанных, независимо развертываемых сервисов. Каждый микросервис инкапсулирует определенную область процесса расчета сплава, например термодинамическое моделирование, расчет фазового равновесия или обработку металлургических данных, работая автономно со специальными базами данных и инфраструктурой.

С технической точки зрения, в этой архитектуре используются принципы проектирования, ориентированные на предметную область, гарантирующие соответствие каждой услуги конкретным бизнес-возможностям. Службы взаимодействуют через облегченные протоколы, такие как RESTful API или асинхронный обмен сообщениями для обмена данными, повышая общую устойчивость и масштабируемость системы.

Приложение для расчета сплавов, вероятно, включает в себя трудоемкие вычислительные задачи, требующие оптимизации производительности микросервисов. Это может включать в себя использование высокопроизводительных вычислительных методов, параллельной обработки и эффективных алгоритмов термодинамических расчетов. Например, генерация фазовой диаграммы обработки микросервиса будет реализовывать алгоритмы вычислительной термодинамики, требующие надежных численных методов для решения сложных фазовых равновесий.

Технологии контейнеризации, такие как Docker, в сочетании с инструментами оркестрации, такими как Kubernetes, являются неотъемлемой частью управления жизненным циклом микросервисов, обеспечения согласованности среды и облегчения непрерывного развертывания и масштабирования. Каждый микросервис разрабатывается, развертывается и масштабируется независимо, что обеспечивает быструю итерацию и гибкое реагирование на меняющиеся требования или научные достижения в металлургии.

Архитектура также включает в себя расширенные возможности обработки данных, использование машинного обучения для прогнозного анализа и технологии больших данных для управления большими наборами данных о металлургических свойствах. Это позволяет приложению предоставлять более точные и основанные на данных прогнозы поведения сплавов в различных условиях.

Подводя итог, можно сказать, что микросервисная архитектура для веб-приложения для расчета сплавов представляет собой технологически продвинутый подход, объединяющий предметно-ориентированное проектирование, контейнеризацию, высокопроизводительные вычисления и анализ данных, чтобы предоставить надежный, масштабируемый и эффективный инструмент для металлургического анализа и инноваций.

2.1 Микросервисная архитектура

Микросервисная архитектура предоставляет ряд ключевых преимуществ, которые делают ее привлекательным выбором для многих проектов. Одним из наиболее значимых преимуществ является масштабируемость. Микросервисы могут масштабироваться независимо друг от друга, что позволяет эффективно управлять нагрузкой и ресурсами. Такой подход обеспечивает гибкость и способствует эффективному использованию ресурсов.

Независимость и гибкость – еще одна важная характеристика микросервисной архитектуры. Каждый микросервис представляет собой отдельный компонент, что обеспечивает независимость разработки, обновления и развертывания. Команды могут

работать параллельно над разными микросервисами, что способствует более быстрой разработке и внедрению новых функций.

Технологическая свобода также является существенным преимуществом микросервисов. Разные микросервисы могут использовать разные технологии и языки программирования в зависимости от их специфических требований. Это позволяет выбирать наилучшие инструменты для каждой конкретной задачи и обеспечивает гибкость в выборе технологических стеков.

Изоляция ошибок – еще одно важное преимущество. Если один микросервис выходит из строя или вызывает ошибку, это не влияет на работу других микросервисов. Изоляция ошибок обеспечивает стабильность работы системы в целом и минимизирует риски.

Легкость развертывания и обновления – еще одно достоинство микросервисов. Изменения в микросервисах можно внедрять более быстро и без остановок системы. Это позволяет ускорить процесс разработки и обновления.

Микросервисная архитектура также облегчает мониторинг и управление системой благодаря изолированной структуре микросервисов и их ограниченной функциональности. Это упрощает выявление и устранение проблем.

Эффективное управление доступностью и отказоустойчивостью – еще одно преимущество. Благодаря изоляции и независимости микросервисов, система может быть более доступной и отказоустойчивой. Микросервисы также хорошо сочетаются с практиками DevOps, так как позволяют автоматизировать процессы развертывания, масштабирования и мониторинга. Микросервисы обеспечивают легкость внедрения новых технологий и инструментов, поскольку их можно внедрять по мере необходимости, без изменения всей системы.

2.2 Описание ключевых компонентов программного продукта и их функциональных ролей

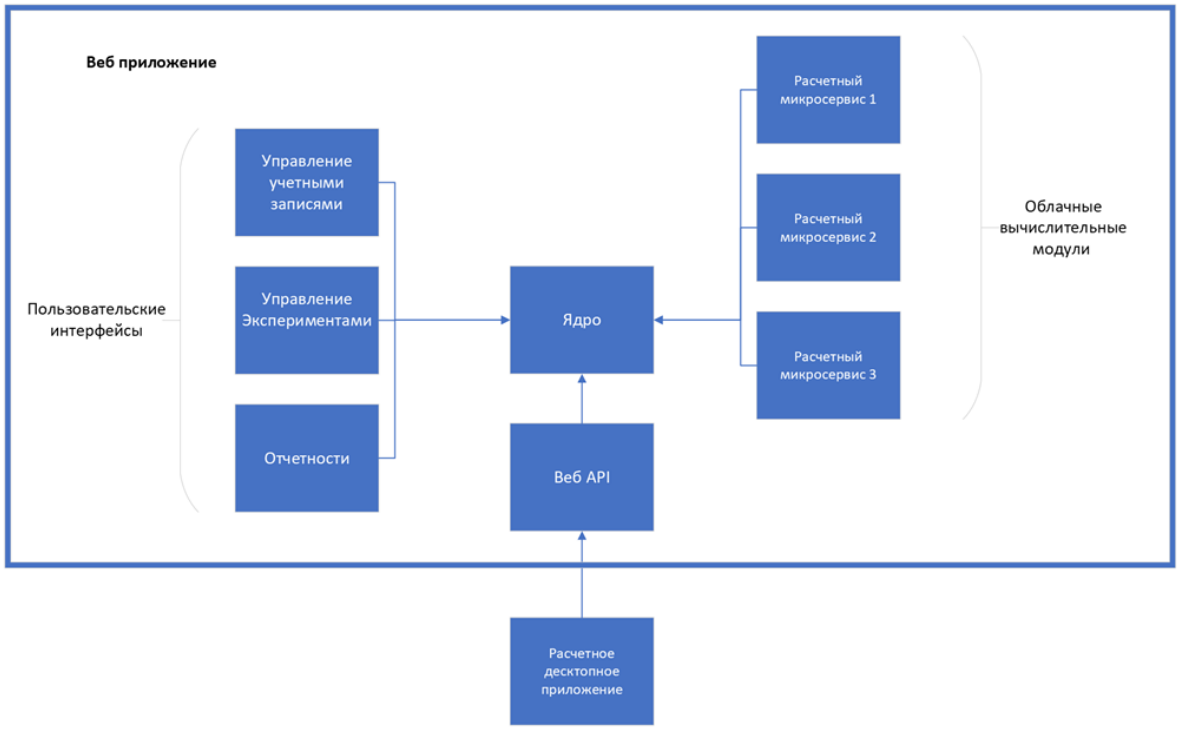

Рисунок 1 - Общая схема работы программного обеспечения

2.3 Проектирование и разработка диаграммы архитектуры, включая взаимосвязи между компонентами

2.3.1 Ядро приложения

Ядро приложения — это центральная часть программного продукта, которая обеспечивает его основную функциональность и базовую архитектуру. Ядро может быть рассмотрено как основа приложения, на которой строятся дополнительные компоненты и функции.

В контексте микросервисной архитектуры, ядро приложения может представлять собой один из микросервисов или набор микросервисов, которые обеспечивают ключевую функциональность системы. Ядро приложения определяет бизнес-логику и правила, по которым работает приложение.

Функциональные характеристики ядра приложения:

1. Основная функциональность: Ядро приложения реализует основные функции, которые обычно выполняет приложение. Например, в случае системы для комплексной переработки металлургического сырья, ядро может содержать логику для обработки данных, оптимизации процессов, расчетов и другие ключевые функции.
2. Бизнес-правила: Ядро приложения содержит бизнес-правила и логику, которая определяет, как приложение должно взаимодействовать с данными и выполнять задачи, связанные с его целью.
3. Взаимодействие с другими компонентами: Ядро может взаимодействовать с другими микросервисами или компонентами приложения, предоставляя им доступ к своей функциональности или получая данные от них.
4. Безопасность и аутентификация: Ядро обычно ответственно за обеспечение безопасности и аутентификации пользователей и других системных компонентов.
5. Масштабируемость и производительность: Ядро приложения должно быть спроектировано с учетом масштабируемости и производительности, чтобы обеспечить эффективную работу системы даже при увеличении нагрузки.
6. Управление данными: Ядро может управлять доступом к данным, их хранением и обработкой.

Ядро приложения представляет собой ключевую часть архитектуры и может взаимодействовать в себя множество компонентов, модулей и сервисов. Он служит основой для дополнительных разработок и расширений.

2.3.2 Пользовательский интерфейс

UI (User Interface) представляет собой интерфейс пользователя в приложении или системе, который позволяет взаимодействовать с программой. Этот интерфейс включает в себя элементы дизайна, контролы (controls) и элементы управления, которые пользователь видит на экране и с которыми взаимодействует.

Важные аспекты UI включают в себя:

1. Графический дизайн. Внешний вид и оформление пользовательского интерфейса, включая цвета, шрифты, изображения и макеты, которые делают приложение удобным и привлекательным для пользователей.
2. Элементы управления. Элементы, с которыми пользователь взаимодействует, такие как кнопки, текстовые поля, списки, выпадающие меню и другие элементы интерфейса.
3. Навигация. Способы, которыми пользователь перемещается по приложению или системе, включая меню, ссылки, вкладки и другие элементы для перехода между разделами и функциональностью.

4. Валидация и обратная связь. Механизмы, которые сообщают пользователю о результате его действий и предотвращают ввод некорректных данных.

5. Реакция на действия пользователя. Как приложение реагирует на действия пользователя, включая анимации, переходы между экранами и изменения состояния элементов интерфейса.

6. Адаптивность и респонсивный дизайн. Умение интерфейса подстраиваться под разные устройства и разрешения экранов, чтобы обеспечить комфортное использование на различных платформах и устройствах.

7. Доступность. Обеспечение доступности интерфейса для пользователей с ограниченными возможностями, такими как люди с инвалидностью.

8. Тестирование и отладка. Проверка работы интерфейса на наличие ошибок и проблем в пользовательском опыте.

UI является важным компонентом успешных приложений, поскольку он оказывает значительное влияние на удовлетворение пользователей и их способность взаимодействовать с программой. Хороший UI помогает сделать приложение интуитивно понятным и удобным в использовании.

2.3.3. Расчетный микросервис

Расчетный микросервис представляет собой независимый компонент системы, который специализируется на выполнении вычислений по конкретному методу или алгоритму. Его основной целью является получение данных, выполнение расчетов и предоставление результатов в удобной форме для ядра системы и других компонентов, которые могут использовать эти данные для дальнейших операций. Основные характеристики и функциональность:

1. Взаимодействие с ядром. Расчетный микросервис устанавливает связь с ядром системы для передачи расчетных данных и получения инструкций.

2. Использование сторонних источников данных. Модуль может обращаться к сторонним источникам данных, если это необходимо для проведения расчетов. Это может включать в себя запросы к базам данных, API или другим сервисам.

3. Выполнение расчетов. Микросервис выполняет расчеты, используя конкретный метод или алгоритм, предназначенный для решения определенной задачи. Расчеты могут быть математическими, статистическими, аналитическими или другого характера, в зависимости от требований.

4. Обработка данных. Полученные результаты обрабатываются и форматируются в соответствии с ожиданиями ядра системы и других компонентов. Это может включать в себя агрегацию данных, преобразование форматов, фильтрацию или другие манипуляции.

5. Предоставление данных. Расчетный микросервис предоставляет готовые результаты в форме, удобной для использования, например, в виде API-эндпоинта, сообщений или других форматов данных.

6. Масштабируемость. Модуль проектируется и разрабатывается с учетом возможности масштабирования, чтобы обеспечивать производительность и эффективность при обработке больших объемов данных или при необходимости параллельного выполнения расчетов.

7. Мониторинг и отслеживание. Для обеспечения надежности и производительности модуль может быть оборудован системой мониторинга и журналирования, которая позволяет отслеживать его работу и выявлять проблемы.

Этот расчетный микросервис играет важную роль в системе, обеспечивая вычислительную мощность и специализированные расчеты для различных аспектов системы. Он может быть интегрирован с другими модулями, которые зависят от его данных и

функциональности, и обеспечивать удовлетворение требований системы к анализу и обработке данных.

2.3.4 Расчетное десктопное приложение

Расчетное десктопное приложение представляет собой полноценное программное решение, специализирующееся на выполнении вычислений по конкретному методу или алгоритму. Это приложение спроектировано для удобства использования на рабочих станциях пользователей и предоставляет возможность взаимодействия с ядром системы и другими компонентами.

Основные характеристики и функциональность:

1. Графический интерфейс. Расчетное десктопное приложение обладает графическим интерфейсом, который обеспечивает удобное взаимодействие с пользователем. Это может включать в себя окна, формы, элементы управления и графические элементы для ввода данных и визуализации результатов.
2. Взаимодействие с ядром системы. Приложение устанавливает связь с ядром системы для передачи данных и получения инструкций. Это включает в себя возможность отправки запросов на расчеты и получения результатов.
3. Использование сторонних источников данных. При необходимости, приложение может обращаться к сторонним источникам данных для получения необходимых входных данных.
4. Выполнение расчетов. Приложение выполняет расчеты, используя конкретный метод или алгоритм, предназначенный для решения определенной задачи. Результаты расчетов могут быть отображены на графическом интерфейсе.
5. Обработка данных и визуализация. Полученные результаты обрабатываются и визуализируются на графическом интерфейсе приложения, что позволяет пользователям легко интерпретировать результаты.
6. Масштабируемость. Приложение проектируется и разрабатывается с учетом возможности масштабирования, чтобы обеспечивать производительность и эффективность при обработке данных.

2.4 Структура базы данных

Для обеспечения гибкости хранения разнообразных данных, используемых в различных методах и вычислениях, предлагается следующая структура базы данных, основанная на системе управления базами данных MySQL.

Нормализация данных. Для обеспечения целостности данных и избежания избыточности информации, данные организованы с использованием принципов нормализации. Это помогает поддерживать базу данных в чистом и структурированном виде. Для хранения различных типов данных, используемых в разных методах вычислений используется подход полиморфизма. Таблицы содержат поля для хранения разных типов данных и дополнительные поля для их идентификации и связи с методами. MySQL предоставляет возможность использования JSON-полей для хранения сложных структур данных. Это позволяет хранить данные в формате JSON и извлекать их по мере необходимости для различных методов вычислений. Имеется возможность использования динамических схем, которые позволяют добавлять и изменять поля и структуры данных по мере необходимости. Для обеспечения хорошей производительности системы уделяется особое внимание оптимизации запросов и настройке индексов в соответствии с требованиями.

Эта структура базы данных позволяет обеспечить гибкое и эффективное управление данными в системе, а также удовлетворить требования по хранению и использованию разнообразных данных в рамках проекта.

2.5 Основные задачи и аспекты интеграции

Архитектура создаваемой микро сервисной системы предусматривает интеграцию с внешними инструментами, сервисами и ресурсами.

Основные задачи и аспекты интеграции включают в себя:

Интеграция с внешними API. Проектирование механизмов для взаимодействия с внешними API, позволяющими получать и отправлять данные между системой и сторонними сервисами. Это может включать в себя интеграцию с внешними поставщиками данных, платежными шлюзами и другими внешними ресурсами.

Интеграция с внешними базами данных. Установление связей и интеграция с базами данных сторонних систем, если это необходимо для доступа к дополнительной информации или данным.

Использование сторонних библиотек и компонентов. Оценка и интеграция сторонних библиотек и компонентов, которые могут улучшить функциональность системы или ускорить разработку.

Обеспечение безопасности. Проектирование механизмов обеспечения безопасности и аутентификации при интеграции с внешними системами, чтобы защитить систему от несанкционированного доступа и утечек данных.

Мониторинг и отслеживание интеграции. Введение мониторинга и журналирования для отслеживания состояния интеграции и выявления проблем в реальном времени.

Тестирование интеграции. Проведение тестирования интеграции с внешними инструментами для уверенности в корректной работе системы и ее способности взаимодействовать с внешними ресурсами.

Интеграция с внешними инструментами является неотъемлемой частью разработки микросервисной системы и обеспечивает ее способность взаимодействовать с внешним миром, расширяя функциональные возможности и обеспечивая более широкий спектр операций.

3 ТЕХНОЛОГИЧЕСКИЕ РЕШЕНИЯ

Система содержит следующие компоненты:
- Сервер ядра;
- Микросервисные вычислительные модули;
- Внешние службы;
- Пользовательский интерфейс (UI) пользователя;
- Управление;
- Общее взаимодействие компонентов.

Сервер ядра - является центральным узлом системы, который обрабатывает бизнес-логику и управляет взаимодействиями между различными компонентами. Он соединен с базой данных, которая хранит всю необходимую информацию для выполнения бизнес-функций. Сервер ядра представляет собой основу, через которую проходят все запросы и данные, обеспечивая централизованное управление и координацию работы системы.

Микросервисные вычислительные модули — это отдельные сервисы, каждый из которых специализируется на определенной вычислительной задаче и имеет собственную базу данных. Данные модули работают независимо от центрального ядра и могут масштабироваться в соответствии с требованиями. Микросервисы оснащены адаптерами входных/выходных данных для обмена информацией с ядром и внешними службами.

Внешние службы — это дополнительные сервисы, которые расширяют функционал системы и предоставляют дополнительные возможности, такие как интеграция с внешними данными, сервисами или API. Данные модули взаимодействуют с микросервисными модулями, обогащая их вычислительные возможности и расширяя область их применения.

Пользовательский интерфейс (UI) пользователя - предоставляет графический интерфейс для взаимодействия с системой. Он позволяет пользователям отправлять запросы, получать данные и управлять процессами в системе. Пользовательский интерфейс может быть реализован через веб-браузер или десктопные приложения и связан с сервером ядра через REST API.

Управление экспериментами -этот компонент включает инструменты и функции для управления экспериментальными данными и процессами. Он обеспечивает интеграцию пользовательского интерфейса с вычислительными процессами и базой данных через REST API, что позволяет пользователям настраивать и контролировать эксперименты.

Общее взаимодействие компонентов - компоненты системы общаются друг с другом с использованием REST API, обеспечивая гибкость и удобство интеграции различных частей системы. REST API является ключевым элементом взаимодействия между клиентскими приложениями и сервером, а также между различными микросервисами и внешними службами.

На рисунке 2 представлена архитектура программного продукта. Данная архитектура обеспечивает модульность, гибкость и масштабируемость, позволяя системе адаптироваться под меняющиеся требования и легко интегрироваться с новыми сервисами и тех.

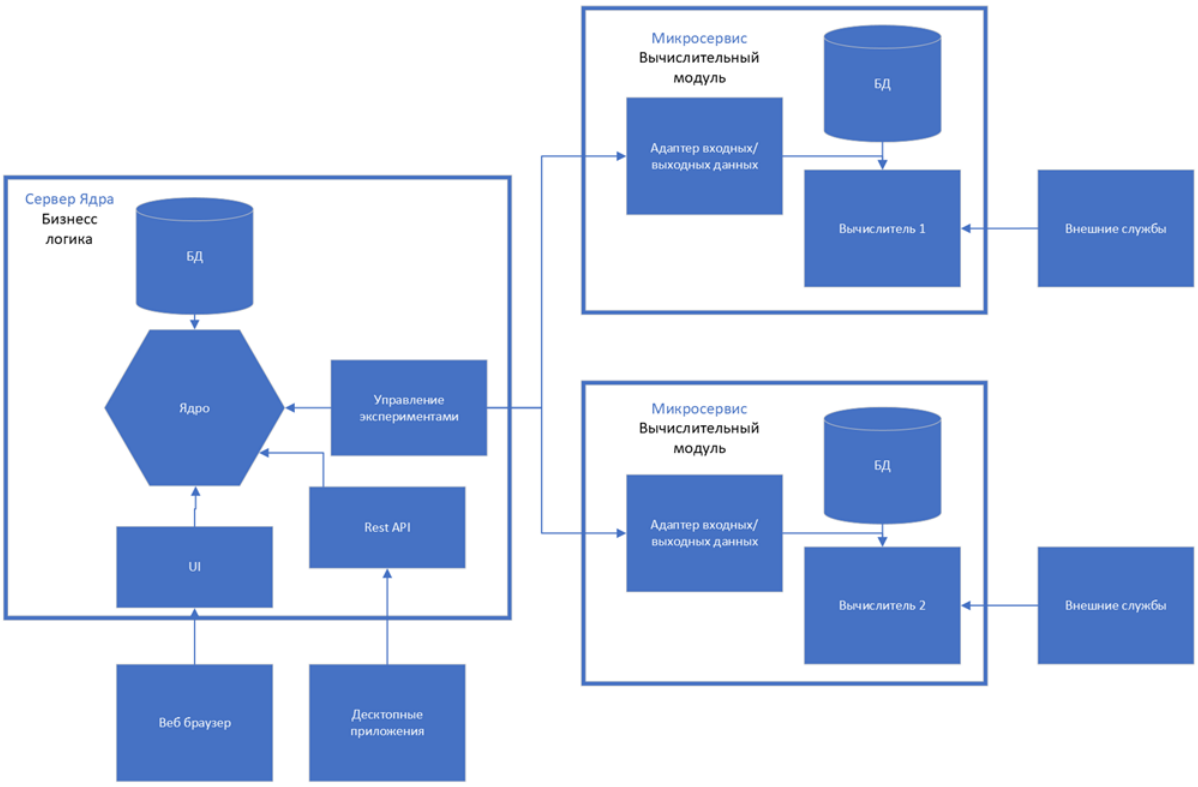

Рисунок 2 - Архитектура системы (в диаграмме отражается микросервисный подход к разработке программного обеспечения)

3.1 Обзор используемых технологий и инструментов

1. ASP.NET Core MVC: язык программирования и фреймворк для создания ядра в предлогаемой системы и веб-приложений.
2. Готовые шаблоны административных панелей: используются для ускорения разработки пользовательских интерфейсов и администрирования.
3. Python: язык программирования для реализации микросервисов, обеспечивающих вычислительные задачи.
4. MySQL: реляционная база данных для хранения данных системы.
5. Docker: технология контейнеризации, используемая для упаковки и публикации приложений и микросервисов данного проекта.

3.2 Обоснование выбора конкретных технологий с учетом требований проекта

Выбор и обоснование технологических решений в проекте представляют собой критически важный этап, требующий обоснованного выбора каждой используемой технологии и инструмента. В данном разделе будут рассмотрены выбранные технологии с учетом их соответствия требованиям проекта.В качестве языка программирования и фреймворка для разработки ядра системы был выбран ASP.NET Core MVC. Этот выбор обоснован его высокой производительностью, масштабируемостью и надежностью, что существенно важно для обработки и анализа больших объемов данных, характерных для проекта. ASP.NET Core MVC также обеспечивает возможность разработки современных веб-приложений с функциональными интерфейсами, что соответствует требованиям проекта.Готовые шаблоны административных панелей: Для ускорения разработки и обеспечения удобства администрирования системы было решено использовать готовые

шаблоны административных панелей. Это обосновано необходимостью сокращения времени разработки и предоставления пользовательских интерфейсов высокого качества для администраторов. В качестве языка программирования для реализации микросервисов был выбран Python. Этот выбор обоснован его гибкостью и богатым набором библиотек, что делает его идеальным для реализации разнообразных вычислительных методов, необходимых для проекта. Python также обладает большой поддержкой и активным сообществом разработчиков, что обеспечивает доступ к решениям и библиотекам, упрощающим разработку. Выбор реляционной базы данных MySQL обоснован ее производительностью, масштабируемостью и надежностью. MySQL предоставляет эффективный механизм хранения данных, что важно для обеспечения надежности и безопасности информации в системе. База данных также поддерживает транзакции, что обеспечивает целостность данных.Docker был выбран для публикации и управления приложениями и микросервисами в виде контейнеров. Это обосновано необходимостью упрощения развертывания и обеспечения изолированности микросервисов. Docker обеспечивает легкость и быстроту развертывания, а также упрощает управление множеством компонентов системы.Выбор каждой из этих технологий был основан на их соответствии требованиям проекта, обеспечивая необходимую производительность, функциональность и удобство разработки. Процесс непосредственной разработки приложения для решения комплексной переработки сложного металлургического сырья требует тщательного выбора инструмента разработки Desktop-приложения и UI/UX дизайна. В реализации перечисленных функций (ввода данных и его визуализации) рассматривается использование фреймворка Electron. Electron предоставляет возможность создавать кроссплатформенные приложения, используя стандартные веб-технологии, такие как HTML, CSS и JavaScript. Это обеспечивает гибкость и универсальность, позволяя запускать приложение на различных операционных системах, таких как Windows, macOS и Linux. Для понимания всех мощности фрейморвка Electron, выделены его основные компоненты:

- Electron включает в себя движок Chromium, который является основой браузера Google Chrome. Это обеспечивает мощный движок для рендеринга веб-страниц и обеспечивает поддержку современных веб-стандартов.

- Electron также включает серверную часть с использованием Node.js. Это позволяет выполнять серверные операции и взаимодействовать с системой, что обеспечивает доступ к низкоуровневым ресурсам компьютера.

- Фреймворк предоставляет API для взаимодействия с операционной системой, файловой системой, работой с окнами приложения, управлением процессами и другими системными функциями.

Для разработки приложения, направленного на переработку металлургического сырья с применением фреймворка Electron, предполагается создавать пользовательский интерфейс с использованием основных веб-технологий. В то же время, взаимодействие с локальной системой и передача данных на сервер будут осуществляться через функционал API Electron.

Помимо, Electron рассматриваются варианты использования нескольких фреймворков Python, которые также могут быть использованы для создания приложения по переработке металлургического сырья.

PyQt/PySide - мощные и гибкие фреймворки для создания кроссплатформенных пользовательских интерфейсов. Поддерживают множество платформ, в том числе Windows, macOS, Linux, iOS и Android. Хорошо подходят для создания приложений с любым уровнем сложности пользовательского интерфейса, включая сложные и анимированные интерфейсы. Хорошо подходят для приложений, требующих взаимодействия с аппаратным обеспечением или другими системами.

Tkinter - стандартная библиотека Python для создания графических интерфейсов. Легковесный, прост в использовании, поставляется вместе с Python. Приложения, написанные на основе данной библиотеки очень быстрые и легковесные.

Kivy - фреймворк для разработки мультимедийных приложений, включая десктопные приложения. Поддерживает множество платформ, в том числе Windows, macOS, Linux, iOS и Android. Хорошо подходит для создания приложений с сенсорным интерфейсом. Поддерживает анимацию и видео. Особенно полезен для создания приложений с более сложными интерфейсами и визуализацией данных. Хорошо подходит для приложений, которые должны быть интерактивными и привлекательными для пользователя.

wxPython - гибкий фреймворк для создания кроссплатформенных пользовательских интерфейсов. Имеет большое количество элементов управления. Поддерживает множество платформ, в том числе Windows, macOS, Linux. Подходит для создания разнообразных десктопных приложений, включая те, которые требуют сложного взаимодействия с пользователем.

3.3 Проектирование архитектурных решений для обеспечения производительности, масштабируемости и безопасности

В данном подразделе приведены выбранные архитектурные решения для обеспечения высокой производительности, способности к масштабированию и обеспечения безопасности данных и приложений.

Производительность. В контексте архитектуры данного проекта было принято решение использовать микросервисную архитектуру. Это решение обеспечивает возможность распределения вычислительной нагрузки между независимыми компонентами системы, что способствует более эффективному использованию ресурсов и обеспечивает высокую производительность. Для обеспечения высокой производительности системы были спроектированы механизмы оптимизации запросов к базе данных. Эти меры включают в себя использование индексов, кэширование и другие методы для минимизации времени обработки данных.

Для операций, которые могут быть выполнены параллельно, было решено использовать асинхронное выполнение, что увеличивает производительность системы и обеспечивает более быстрые ответы на запросы.

Масштабируемость. В контексте архитектуры была предусмотрена возможность горизонтального масштабирования. Это решение позволяет добавлять новые серверы и ресурсы для равномерного распределения нагрузки при увеличении объемов данных или запросов.

Для обеспечения масштабируемости компонентов системы было решено использовать контейнеризацию. Это позволяет упаковывать и разворачивать микросервисы независимо друг от друга, облегчая масштабирование.

4 ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММНОГО ОБЕСПЕСЕНИЯ

4.1 Вопросы безопасности и отладки программного обеспечения

Безопасность кода. Архитектура включает в себя регулярную проверку кода на предмет уязвимостей, таких как инъекции и переполнения буфера. Это обеспечивает высокий уровень безопасности кода.

Архитектурные решения предусматривают внедрение механизмов аутентификации и авторизации, ограничивая доступ к системе только уполномоченными пользователями и храня пароли в зашифрованном виде.

В архитектуре предусмотрен мониторинг безопасности, который позволяет быстро обнаруживать и реагировать на потенциальные угрозы и атаки.

Качество кода. Архитектурные решения следуют строгим стандартам кодирования, обеспечивая чистоту и читаемость кода. Это включает в себя структуру проекта, именование переменных и комментирование кода.

Все компоненты кода подвергаются тестированию, включая юнит-тесты, интеграционные и функциональные тесты, что помогает выявить и устранить ошибки на ранних этапах разработки.

Для обеспечения стабильности и надежности кода в архитектуре внедрена автоматизация сборки и развертывания приложений с использованием средств CI/CD.

4.2 Оптимизация производительности и выявление узких мест

Уделено внимание обеспечению высокой производительности системы и выявлению узких мест в процессе ее функционирования. Для достижения этой цели внедрены следующие стратегии:

Осуществляется профилирование кода и запросов для выявления узких мест. Результаты профилирования используются для оптимизации кода и запросов к базе данных, что способствует улучшению производительности.

Предусмотрена возможность добавления дополнительных ресурсов, включая серверы, при выявлении узких мест. Это позволяет масштабировать систему и обеспечивать ее эффективную работу.

Результаты мониторинга приводят к постановке задач на устранение или оптимизацию выявленных узких мест, и процесс оптимизации производительности становится непрерывным, включая анализ, оптимизацию и проверку результатов.

Эти архитектурные решения в целом спроектированы для обеспечения высокой производительности системы и оперативного выявления, а также устранения узких мест, что гарантирует эффективную работу приложений и сервисов.

ЗАКЛЮЧЕНИЕ

Архитектурная концепция, описанная в данном документе, представляет собой фундаментальную основу для разработки программного продукта, предназначенного для комплексной переработки сложного металлургического сырья. Данная архитектура была разработана с учетом основных требований проекта и целей, которые включают в себя обеспечение высокой производительности, масштабируемости, безопасности и качества кода.

Микросервисная архитектура выбрана в качестве основного структурного решения, позволяя создать независимые модули, способные использовать различные технологии, данные и инструменты. Это обеспечивает гибкость и масштабируемость системы, а также позволяет интегрировать множество вычислительных методов и внешних ресурсов.

Выбор технологий, таких как ASP.NET Core MVC, Python и MySQL, основывается на требованиях проекта и удовлетворяет его потребности в эффективной разработке и вычислениях. Применение контейнеризации с использованием Docker обеспечивает удобство развертывания и масштабирования микросервисов.

Безопасность и качество кода являются неотъемлемой частью архитектурного подхода. Регулярная проверка на уязвимости, аутентификация, авторизация, мониторинг и логирование обеспечивают безопасность системы. Стандарты кодирования, тестирование и код-ревью гарантируют качество кода.

Оптимизация производительности и выявление узких мест становятся непрерывным процессом благодаря мониторингу, профилированию и анализу результатов. Это обеспечивает бесперебойную и эффективную работу системы.

В целом, архитектурная концепция, описанная в данном документе, представляет собой комплексный подход к разработке программного продукта для комплексной переработки металлургического сырья, обеспечивая высокую производительность, масштабируемость, безопасность и качество, что делает ее надежной и перспективной основой для будущего развития проекта.

В рамках проекта была успешно создана архитектура надежной вычислительная системы, составляющая ядро архитектуры программного обеспечения данного проекта. Эта архитектура разработана для выполнения сложных расчетов, необходимых в металлургических процессах.

Был использован широкий спектр алгоритмов, каждый из которых адаптирован к различным аспектам металлургической обработки. Такое разнообразие обеспечивает всесторонний охват всех аспектов процесса, включая термодинамические расчеты, химическую кинетику и оценку свойств материалов.

Эмпирические данные различных металлургических процессов были эффективно интегрированы, что повысило точность и надежность расчетов данного проекта. Этот подход, основанный на данных, позволил получить более точные прогнозы и модели, что имеет решающее значение для эффективной оптимизации процессов.

Ключевой особенностью были адаптивные алгоритмы, предназначенные для повышения точности и эффективности по мере обработки большего количества данных. Такая адаптивность была необходима для долгосрочной оптимизации процессов и постоянного улучшения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Mineral processing equipment - https://cogentdatahub.com/mineral-processing-equipment/
2. Jacobus Willem De Villiers Groenewald. A Process Performance Monitoring Methodology for Mineral Processing Plants. PhD thesis, 2014.
3. 2021 - AVEVA PI World 2021 - Manufacturing; Mining & Metals; Pulp & Paper - https://resources.osisoft.com/presentations/merian-production-reporting/
4. Kithulwatta, W.M.C.J.T. & Jayasena, K.P.N & Kumara, Banage & Rathnayaka, R.M. Kapila. (2022). Integration With Docker Container Technologies for Distributed and Microservices Applications: A State-of-the-Art Review. International Journal of Systems and Service-Oriented Engineering. 10.4018/IJSSOE.297136.
5. Distributed Microservice Architecture with Docker Imanol Urra Ruiz. Master thesis, 2016.
6. Tapia, F., Mora, M. Á., Fuertes, W., Aules, H., Flores, E., & Toulkeridis, T. (2019). From Monolithic Systems to Microservices: A Comparative Study of Performance. *Applied Sciences*, *10*(17), 5797. https://doi.org/10.3390/app10175797
7. Saboor, A., Hassan, M. F., Akbar, R., Shah, S. N., Hassan, F., Magsi, S. A., & Siddiqui, M. A. (2021). Containerized Microservices Orchestration and Provisioning in Cloud Computing: A Conceptual Framework and Future Perspectives. *Applied Sciences*, *12*(12), 5793. https://doi.org/10.3390/app12125793
8. Psomakelis, E., Nikolakopoulos, A., Marinakis, A., Psychas, A., Moulos, V., Varvarigou, T., & Christou, A. (2020). A Scalable and Semantic Data as a Service Marketplace for Enhancing Cloud-Based Applications. *Future Internet*, *12*(5), 77. https://doi.org/10.3390/fi12050077
9. Kazanavičius, J., Mažeika, D., & Kalibatienė, D. (2021). An Approach to Migrate a Monolith Database into Multi-Model Polyglot Persistence Based on Microservice Architecture: A Case Study for Mainframe Database. *Applied Sciences*, *12*(12), 6189. https://doi.org/10.3390/app12126189
10. Manuel, J., & Sención, G. (2020). Evaluating Service-Oriented and Microservice Architecture Patterns to Deploy eHealth Applications in Cloud Computing Environment. *Applied Sciences*, *11*(10), 4350. https://doi.org/10.3390/app11104350
11. Tsepelev, V. S., Starodubtsev, Y. N., & Tsepeleva, N. P. (2021). Thermophysical Properties of Pipe Steel in the Liquid State. *Metals*, *11*(7), 1099. https://doi.org/10.3390/met11071099
12. Liu, X., & Aldrich, C. (2022). Assessing the Influence of Operational Variables on Process Performance in Metallurgical Plants by Use of Shapley Value Regression. Metals, 12(11), 1777. https://doi.org/10.3390/met12111777
13. Jančar, D., Machů, M., Velička, M., Tvardek, P., & Vlček, J. (2022). Use of Numerical Methods for the Design of Thermal Protection of an RFID-Based Contactless Identification System of Ladles. Metals, 12(7), 1163. https://doi.org/10.3390/met12071163
14. Liu, X., & Aldrich, C. (2022). Deep Learning Approaches to Image Texture Analysis in Material Processing. Metals, 12(2), 355. https://doi.org/10.3390/met12020355
15. Jawahery, S., Visuri, V., Wasbø, S. O., Hammervold, A., Hyttinen, N., & Schlautmann, M. (2021). Thermophysical Model for Online Optimization and Control of the Electric Arc Furnace. *Metals*, *11*(10), 1587. https://doi.org/10.3390/met11101587
16. Zhang, C., Jiang, X., Zhang, R., Wang, X., Yin, H., Qu, X., & Liu, Z. (2019). High-throughput thermodynamic calculations of phase equilibria in solidified 6016 Al-alloys. *Computational Materials Science*, *167*, 19-24. https://doi.org/10.1016/j.commatsci.2019.05.022
17. Kong, H., Hou, Y., Qin, H., Xie, J., Bi, Z., & Su, H. (2023). Integrated Thermal and Metallurgical Simulation Framework for Selective Laser Melting Multi-Component and Multi-Phase Alloys. *Processes*, *11*(12), 3289. https://doi.org/10.3390/pr11123289.

ПРИЛОЖЕНИЕ 1

1. Функциональные требования:
   1.1. Программный продукт должен предоставлять средства для проведения вычислений и анализа данных, связанных с металлургической и обогатительной промышленностью.
   1.2. Продукт должен поддерживать различные вычислительные методы и алгоритмы, необходимые для обработки сложных данных.
   1.3. Пользовательский интерфейс продукта должен быть интуитивно понятным и удобным для пользователей, включая возможность визуализации результатов вычислений.
   1.4. Программный продукт должен обеспечивать возможность интеграции с внешними ресурсами, такими как сторонние базы данных, библиотеки и инструменты.

2. Требования к производительности:
   2.1. Программный продукт должен обеспечивать высокую скорость выполнения вычислений и обработки данных для эффективного использования времени и ресурсов.
   2.2. Продукт должен быть масштабируемым и способным обрабатывать большие объемы данных при необходимости.

3. Требования к безопасности:
   3.1. Программный продукт должен обеспечивать механизмы аутентификации и авторизации пользователей, чтобы предотвратить несанкционированный доступ.
   3.2. Данные, хранящиеся и обрабатываемые в продукте, должны быть защищены от утечек и несанкционированного доступа.

4. Требования к качеству кода:
   4.1. Код продукта должен соответствовать строгим стандартам кодирования для обеспечения его чистоты и читаемости.
   4.2. Продукт должен проходить тестирование на различных уровнях, включая юнит-тестирование и интеграционное тестирование, для выявления и устранения ошибок.
   4.3. Код должен проходить процесс код-ревью, чтобы обеспечить его качество и безопасность.

5. Требования к масштабированию:
   5.1. Продукт должен поддерживать горизонтальное масштабирование для обеспечения равномерного распределения нагрузки.

6. Требования к интеграции:
   6.1. Продукт должен обеспечивать интеграцию с различными внешними инструментами и ресурсами, используемыми в исследованиях и вычислениях.