



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана
(национальный исследовательский университет)» (МГТУ
им. Н.Э. Баумана)

Московский техникум космического приборостроения

Отделение Информационные системы и программирование

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

по профессиональному

модулю ПМ.02 Структурирование информации программных модулей

Код, Специальность 09.02.07. Информационные системы и программирование

Место прохождения практики АО «Корпорация „Космос“
(полное название организации)

Выполнил

студент Владимиров Иван Сергеевич
(фамилия, имя, отчество)

Курс 3 Группа ПИП-62

Подпись студента В

Оценка отлично

Дата приема отчета 24 июля 202 4 г.

Руководитель практики от техникума Беликова О.В.
(подпись) (фамилия, имя, отчество)

202 4

Московский техникум космического приборостроения МГТУ имени Н.Э. Баумана

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ на прохождение производственной практики

на предприятии ООО Корпорация "Космос"

по: ПП 02 ПМ. 02 Осуществление интеграции программных модулей в объеме
144 часов

Студент Владимир Иван Сергеевич, 09.02.04, ММТ-62
(фамилия, имя, отчество; индекс специальности, группа)

Студент во время прохождения производственной практики с 28.05 2024 г. по 24.06. 2024 г.
должен:

1. Ознакомиться:

- с моделями процесса разработки программного обеспечения;
- основными принципами процесса разработки программного обеспечения;
- основными подходами к интегрированию программных модулей;
- основами верификации и аттестации программного обеспечения.

2. Уметь:

- использовать выбранную систему контроля версий;
- использовать методы для получения кода с заданной функциональностью и степенью качества.

3. Получить практический опыт:

- в интеграции модулей в программное обеспечение;
- в отладке программных модулей.

Дата выдачи задания « 28 » 05 2024 г.

Руководитель практики от техникума

Студент

Бел (подпись, дата) Белых А.С. (фамилия И.О.)
В (подпись, дата) Владимир И.С. (фамилия И.О.)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 Постановка задачи.....	6
2 Описание предметной области	8
3 Моделирование проектируемой системы	12
4 Функциональные требования к системе	16
5 Обоснование выбора средств реализации.....	22
6 Проектирование программной системы	25
7 Проектирование программной системы	27
7.1 Схема алгоритма получения цвета по приоритету	27
7.2 Схема алгоритма получения иконки устройства	28
8 Проектирование программной системы	29
9 Руководство пользователя.....	37
ЗАКЛЮЧЕНИЕ	42
СПИСОК ЛИТЕРАТУРЫ.....	44
ПРИЛОЖЕНИЯ.....	46
ПРИЛОЖЕНИЕ А	47
А.1 Листинг файла main.go.....	48
А.2 Листинг файла login.html	68
А.3 Листинг файла main.html	69
А.4 Листинг файла auth.js	71
А.5 Листинг файла styleLogin.css.....	72
А.6 Листинг файла styleMain.css.....	74
А.7 Листинг файла script.js	79
ПРИЛОЖЕНИЕ Б.....	92

ВВЕДЕНИЕ

С развитием информационных технологий и увеличением цифровизации бизнес-процессов важность надежного и эффективного управления информационной инфраструктурой организаций становится все более очевидной. Одним из ключевых аспектов этого процесса является мониторинг и контроль состояния программного обеспечения в локальных сетях.

Существующие методы мониторинга и контроля зачастую ограничены в функциональности и не всегда обеспечивают полный и своевременный анализ состояния программного обеспечения. Это может привести к непредвиденным сбоям в работе, потере данных или даже уязвимостям в системе безопасности.

В условиях стремительно развивающегося информационного общества и повсеместного использования информационных технологий, важность мониторинга и управления состоянием компьютерных систем и сетей становится все более значимой. Системы мониторинга, такие как Zabbix, позволяют обеспечивать бесперебойную работу инфраструктуры, своевременно выявляя и устраняя возможные неисправности. Данный проект посвящен разработке пользовательского интерфейса для мониторинга состояния устройств с использованием Zabbix, что значительно упрощает задачу отслеживания и анализа данных.

Целью данной работы является создание интерактивного веб-интерфейса, который позволит администраторам эффективно отслеживать состояние устройств в сети, оперативно реагировать на возникшие проблемы и получать всю необходимую информацию в удобном виде. Данный веб-интерфейс обеспечит непрерывным мониторингом и контролем состояния программного обеспечения внутри локальных сетей на языке программирования Go в операционной системе Astra Linux. Выбор операционной системы Astra Linux обусловлен необходимостью соответствия требованиям безопасности и надежности проекта. Также учитывается совместимость данной операционной системы с используемыми в АО «Корпорация «Комета» технологиями и инструментами разработки, что обеспечивает эффективную и безопасную работу в рамках проекта.

Для достижения поставленной цели были определены следующие задачи:

- разработка функционала для отображения состояния устройств на основе данных, полученных из системы Zabbix;
- реализация возможности просмотра детальной информации о каждом устройстве;
- обеспечение удобного отображения и хранения данных о текущих триггерах и инцидентах;
- введение механизма уведомлений о неактивных агентах и иных критических состояниях;
- создание функции сохранения данных о состоянии устройств в формате, удобном для дальнейшего анализа и отчётности;
- обеспечение возможности скачивания и открытия отчётов в формате Word, что позволяет легко интегрировать полученные данные в рабочий процесс и документацию.

Таким образом, данный проект направлен на улучшение процесса мониторинга ИТ-инфраструктуры, что позволит снизить время на реакцию и устранение проблем, повысив общую эффективность управления системой.

Данное приложение будет предоставлять возможность оперативно обнаруживать проблемы и сбои в работе программного обеспечения, анализировать эффективность и производительность приложений, а также предупреждать об угрозах безопасности и потенциальных уязвимостях.

1 Постановка задачи

Назначение разрабатываемой программы:

Целью разрабатываемой программы является создание программного комплекса для контроля состояния технических средств, который обеспечит оперативный мониторинг и управление ИТ-инфраструктурой. Комплекс включает в себя веб-сервер для обработки информации о состоянии оборудования и веб-интерфейс для отображения этой информации в удобной и наглядной форме. Основной задачей программы является повышение эффективности мониторинга и управления техническими средствами, минимизация времени реакции на сбои и обеспечение бесперебойной работы системы.

Для достижения поставленных целей необходимо решить следующие задачи:

1) Разработка веб-сервера:

- Создание сервера, который будет получать и обрабатывать данные от системы мониторинга Zabbix.
- Обеспечение возможности хранения и актуализации данных о состоянии технических средств.
- Реализация API для взаимодействия с веб-интерфейсом, включающего функции для получения информации о состоянии устройств и их триггерах.

2) Создание веб-интерфейса:

- Разработка динамически генерируемой интерактивной мнемосхемы оборудования, которая будет отражать текущее состояние технических средств.
- Реализация механизма отображения текущего состояния оборудования с различной цветовой индикацией:
 - Желтым цветом — при некритических сбоях.
 - Красным цветом — при критических сбоях.
 - Зеленым цветом — когда все хорошо.
- Обеспечение возможности запроса подробной информации о любом оборудовании при клике на его пиктограмму.

3) Интерактивность и визуализация:

- Реализация функции отображения всплывающих уведомлений о неактивных агентах и иных критических состояниях.
- Создание механизма обновления состояния пиктограмм оборудования в реальном времени.

4) Сохранение и экспорт данных:

- Введение функции сохранения данных о состоянии оборудования и триггерах в файл формата JSON.
- Реализация возможности экспорта и открытия отчётов в формате Word с определённым форматированием.

5) Обработка ошибок и уведомления:

- Внедрение механизма обработки ошибок при получении данных от Zabbix и отображение соответствующих уведомлений оператору.
- Реализация уведомлений о недоступности агента, когда оборудование отключено или не отвечает.

В результате выполнения поставленных задач должен быть создан программный комплекс, который обеспечит:

- Эффективный и наглядный мониторинг состояния технических средств.
- Быструю реакцию на изменения состояния оборудования.
- Удобный интерфейс для получения детальной информации о каждом устройстве.
- Возможность хранения и анализа данных о состоянии системы.
- Поддержку экспорта данных в удобный формат для дальнейшего использования в документации и отчетах.

2 Описание предметной области

Контроль состояния технических средств является одной из ключевых задач в управлении информационными системами. Современные ИТ-инфраструктуры состоят из множества различных компонентов, включая серверы, рабочие станции, сетевое оборудование и другие устройства, которые необходимо непрерывно мониторить для обеспечения их бесперебойной работы. Разработка программного комплекса для контроля состояния технических средств направлена на повышение эффективности мониторинга и управления ИТ-инфраструктурой, что является актуальной задачей в условиях растущих требований к надежности и доступности информационных систем.

До внедрения специализированного программного комплекса для мониторинга состояния технических средств, управление ИТ-инфраструктурой часто осуществлялось с использованием разрозненных инструментов и методов. Это могло включать ручной сбор данных, использование базовых средств мониторинга, предоставляемых операционными системами или отдельными устройствами, а также применение различных независимых систем мониторинга, которые не были интегрированы друг с другом. Такой подход имел ряд недостатков:

1. Фрагментированность данных:
 - Отсутствие централизованного хранилища данных о состоянии оборудования затрудняло их анализ и принятие решений.
 - Необходимость использования различных интерфейсов для получения информации о состоянии различных устройств.
2. Задержки в реакции на инциденты:
 - Ручной сбор и обработка данных приводили к задержкам в выявлении и устранении проблем.
 - Отсутствие автоматизированных уведомлений о сбоях требовало постоянного внимания со стороны операторов.
3. Отсутствие единой визуализации:

- Необходимость использования нескольких инструментов для мониторинга различных аспектов инфраструктуры усложняла создание общей картины состояния системы.
- Отсутствие интерактивных визуализаций затрудняло выявление взаимосвязей между различными компонентами системы.

Управление сложными IT-инфраструктурами требует решения ряда специфических задач:

1. Сбор и анализ данных:

- Необходимость сбора данных о состоянии различных устройств в реальном времени.
- Анализ полученных данных для выявления проблемных зон и прогнозирования потенциальных сбоев.

2. Обеспечение надежности и доступности:

- Минимизация времени простоя оборудования и предотвращение потерь данных.
- Обеспечение быстрого восстановления работоспособности в случае возникновения сбоев.

3. Централизованное управление:

- Консолидация информации о состоянии всех компонентов IT-инфраструктуры в едином интерфейсе.
- Обеспечение возможности оперативного реагирования на инциденты.

4. Визуализация и уведомления:

- Создание наглядных и интерактивных визуализаций состояния оборудования.
- Автоматизированные уведомления о сбоях и изменениях состояния оборудования.

До внедрения программного комплекса, процесс мониторинга и управления состоянием технических средств можно представить следующим образом:

1. Сбор данных:

- Данные о состоянии оборудования собираются с использованием встроенных средств мониторинга операционных систем и отдельных инструментов для каждого типа устройств.
 - Сбор данных осуществляется вручную или с помощью независимых систем мониторинга.
2. Анализ данных:
- Анализ данных выполняется вручную или с использованием базовых аналитических инструментов.
 - Выявление проблем и прогнозирование сбоев требует значительных временных и человеческих ресурсов.
3. Управление инцидентами:
- В случае выявления проблем, операторы самостоятельно принимают решения о необходимых действиях для устранения сбоев.
 - Время реакции на инциденты зависит от квалификации операторов и их способности оперативно анализировать информацию.
4. Визуализация:
- Визуализация данных о состоянии оборудования осуществляется с использованием различных интерфейсов и инструментов.
 - Отсутствие единой визуализации затрудняет получение общей картины состояния системы.

Внедрение программного комплекса для контроля состояния технических средств направлено на решение вышеуказанных проблем и оптимизацию процессов мониторинга и управления ИТ-инфраструктурой. Программный комплекс включает в себя следующие компоненты:

1. Веб-сервер:
 - Веб-сервер получает данные от системы мониторинга Zabbix, обрабатывает их и хранит в централизованной базе данных.
 - Обеспечивает доступ к данным через API, позволяя веб-интерфейсу запрашивать и отображать актуальную информацию о состоянии оборудования.
2. Веб-интерфейс:

- Веб-интерфейс предоставляет оператору интерактивную мнемосхему оборудования, которая динамически обновляется в зависимости от текущего состояния технических средств.
- Оператор может получать подробную информацию о каждом устройстве, а также просматривать уведомления о сбоях и изменениях состояния оборудования.

3. Уведомления и визуализация:

- Программный комплекс обеспечивает автоматическое уведомление операторов о сбоях и критических изменениях состояния оборудования.
- Визуализация состояния оборудования осуществляется с использованием цветовой индикации, позволяя быстро идентифицировать проблемные устройства.

Внедрение программного комплекса для контроля состояния технических средств позволит значительно повысить эффективность управления ИТ-инфраструктурой, обеспечить её надёжность и доступность, а также минимизировать время реакции на инциденты. Централизованное хранение и анализ данных, интерактивные визуализации и автоматизированные уведомления создадут условия для более оперативного и качественного мониторинга состояния технических средств.

3 Моделирование проектируемой системы

Проектирование и моделирование системы контроля состояния технических средств предполагает использование структурных и функциональных моделей для детального описания всех процессов и элементов системы. Для данной задачи наиболее подходящей методологией является использование диаграмм моделирования бизнес-процессов (IDEF0) и диаграмм вариантов использования (Use Case), которые позволяют наглядно представить взаимодействие различных компонентов системы и пользователей.

На рисунке 3.1 представлена контекстная диаграмма IDEF0, описывающая процесс взаимодействия с Zabbix сервером, взаимодействие с устройствами.

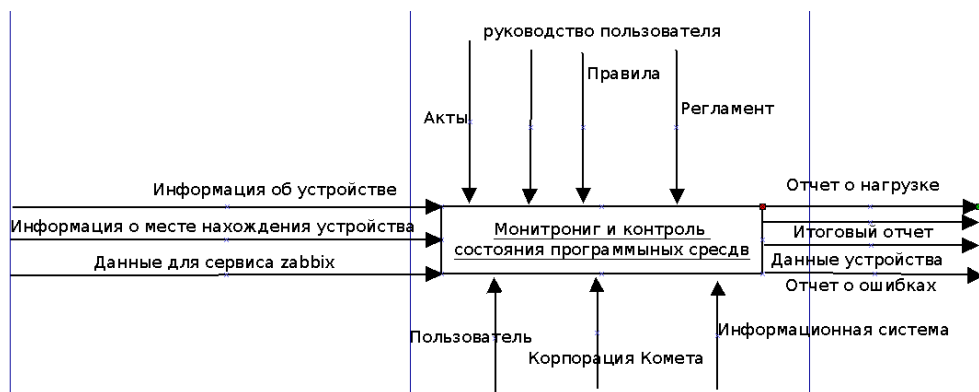


Рисунок 3.1 – Контекстная диаграмма IDEF0

На рисунке 3.2 представлена диаграмма декомпозиции IDEF0

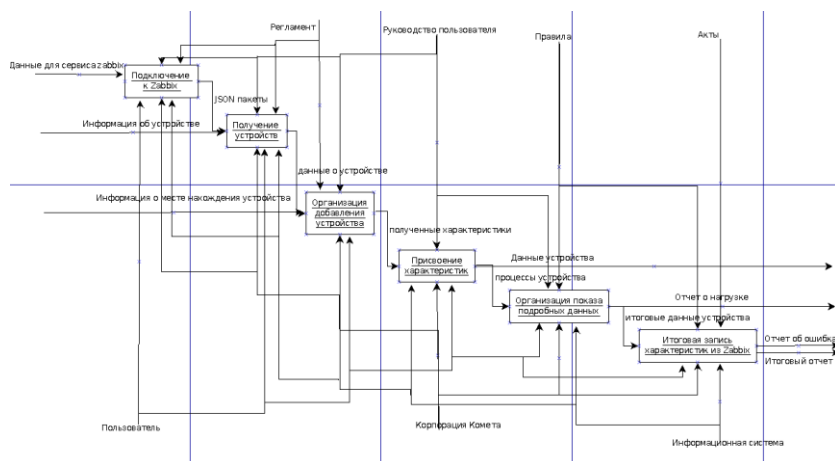


Рисунок 3.2 – Декомпозиция контекстной диаграммы

Входными данными является информация об устройстве, поступающая из Zabbix, информация о месте нахождения устройства и сами данные zabbix. Механизмом деятельности считается разработанное приложение. После обращения пользователя происходит сбор и обработка данных. Данные устройства берутся из сервиса zabbix, также оттуда берутся данные об ошибках, которые выдает нам триггер, и отчет о нагрузке. В самом конце формируется итоговый отчет.

На рисунке 3.3 представлена диаграмма вариантов использования.

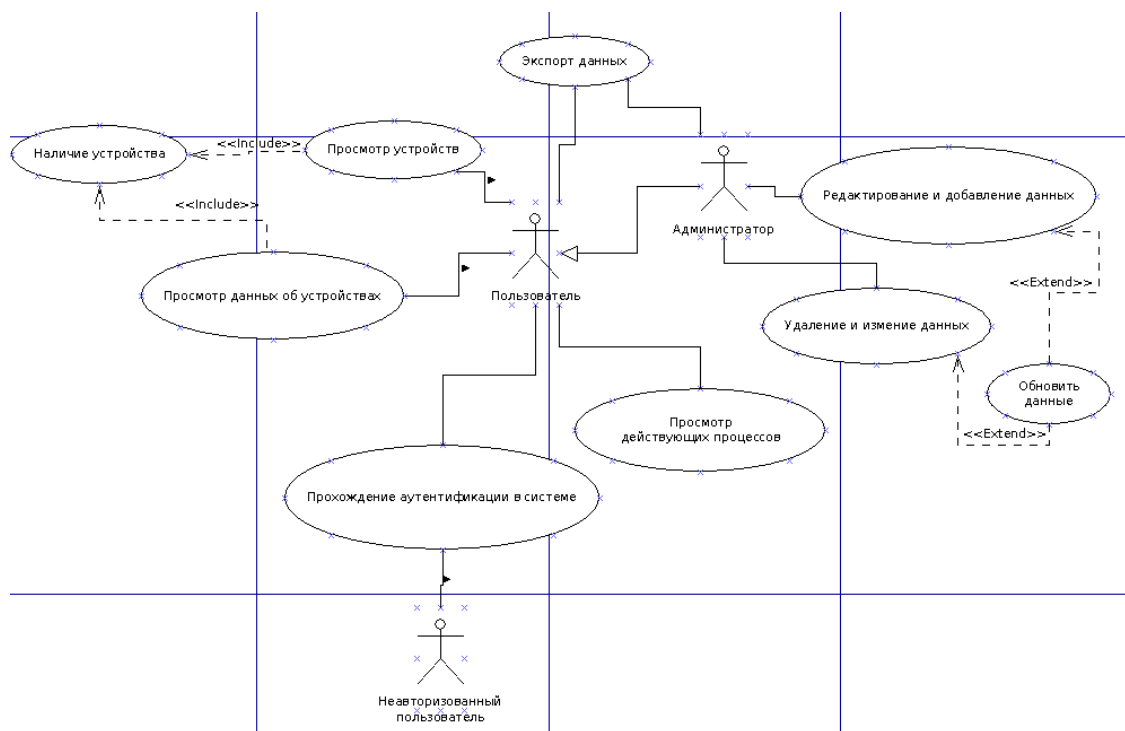


Рисунок 3.3 – Диаграмма вариантов использования

На диаграмме вариантов использования представлены актеры: пользователь, администратор, неавторизованный пользователь. Пользователь проходит аутентификацию (связь - ассоциация), после аутентификацию пользователь может взаимодействовать с системой, он может экспортировать данные (связь – ассоциативная), просматривать устройства и их данные (связь - ассоциация), оно же включает в себя проверку на наличие устройств (связь – включение include), пользователем является и администратор, он может редактировать и удалять данные, это включает в себя обновление данных(связь – расширение extend).

Актеры:

1. Пользователь (Actor):

Оператор или администратор, прошедший аутентификацию в системе.

2. Администратор (Actor):

Пользователь с полными правами доступа, включая управление данными и пользователями.

3. Неавторизованный пользователь (Actor):

Пользователь, не прошедший аутентификацию в системе.

Варианты использования:

Аутентификация (Use Case):

Актер: Неавторизованный пользователь

Описание: Неавторизованный пользователь проходит процесс аутентификации для получения доступа к системе.

Экспорт данных (Use Case):

Актер: Пользователь

Описание: Пользователь имеет возможность экспортировать данные о состоянии технических средств в различные форматы для внешнего использования.

Просмотр устройств и их данных (Use Case):

Актер: Пользователь

Описание: Пользователь может просматривать информацию о состоянии устройств, полученную из системы мониторинга Zabbix.

Проверка наличия устройств (Use Case):

Актер: Пользователь

Описание: Пользователь может проверять наличие устройств в системе и их текущее состояние.

Редактирование данных (Use Case):

Актер: Администратор

Описание: Администратор может редактировать данные о состоянии устройств и другую информацию в системе.

Удаление данных (Use Case):

Актер: Администратор

Описание: Администратор имеет возможность удалять данные о состоянии устройств и другую информацию из системы.

Генерация отчетов (Use Case):

Актер: Пользователь

Описание: Пользователь может создавать отчеты о состоянии технических средств на основе имеющихся данных.

Управление пользователями (Use Case):

Актер: Администратор

Описание: Администратор управляет пользователями системы, включая добавление, редактирование и удаление пользователей, а также назначение ролей.

Описание взаимодействий:

Неавторизованные пользователи проходят аутентификацию для доступа к системе.

Зарегистрированные пользователи (пользователи и администраторы) могут взаимодействовать с системой, выполняя различные действия в зависимости от своих ролей.

Пользователи просматривают, редактируют и удаляют данные о состоянии устройств, а также генерируют отчеты.

Администраторы осуществляют управление пользователями, включая управление доступом и данными.

4 Функциональные требования к системе

Функциональное назначение:

Разрабатываемая программа предназначена для автоматизации процесса мониторинга состояния технических средств и генерации отчетов на основе данных, полученных из системы мониторинга Zabbix. Она обеспечивает следующие функциональные возможности:

- Управление заказами: Создание, редактирование и удаление заказов на техническое оборудование.
- Мониторинг состояния устройств: Автоматический сбор данных о состоянии устройств из системы мониторинга Zabbix.
- Формирование отчетов: Генерация отчетов о состоянии оборудования, нагрузке и ошибках для оперативного анализа и принятия управленческих решений.
- Автоматическое создание документов: Создание документации по заказам и отчетам для клиентов и внутренних пользователей.

Эксплуатационное назначение:

Программа выполняется ежедневно в рабочее время с целью обеспечения текущей оперативной работы и планирования ремонтов и технического обслуживания оборудования. Результаты работы программы используются следующими службами и отделами:

Техническая поддержка:

Получает данные о текущем состоянии оборудования для оперативного реагирования на инциденты и планирования технических вмешательств.

Отдел продаж:

Использует информацию о заказах для учета продаж и планирования поставок оборудования клиентам.

Администрация:

Получает отчеты о нагрузке на оборудование и текущем состоянии системы для управленческого анализа и принятия стратегических решений.

Клиенты:

Используют отчеты и документацию по заказам для контроля статуса своих заказов и получения информации о технических характеристиках и состоянии оборудования.

Программа ориентирована на обеспечение эффективного управления заказами и обслуживания технического оборудования, повышение операционной эффективности и улучшение качества обслуживания клиентов.

Требования к программе

1. Требования к функциональным характеристикам

В этом разделе определяются основные функциональные возможности программы, необходимые для достижения её целей.

Состав выполняемых функций:

- Программа должна автоматически собирать данные о состоянии устройств из системы мониторинга Zabbix.
- Должна быть реализована возможность анализа данных и формирования отчетов на основе полученной информации.
- Программа должна поддерживать экспорт данных в формат DOC для обеспечения удобства анализа и представления результатов.

Организация входных и выходных данных:

- Входные данные должны поступать из системы мониторинга Zabbix посредством соответствующих API запросов.
- Выходные данные, такие как отчеты и уведомления, должны быть организованы в удобном формате для визуализации и передачи заказчику.

Временные характеристики:

- Программа должна оперативно обрабатывать данные и формировать отчеты в заданные сроки, например, каждый час или ежедневно в определенное время.
- Время ответа на запросы должно быть минимальным для обеспечения оперативного мониторинга состояния оборудования.

2. Требования к надежности

- Этот раздел описывает условия и требования, необходимые для обеспечения надежного функционирования программы.
- Обеспечение устойчивого функционирования:

- Программа должна обеспечивать стабильную работу в течение всего периода её использования без сбоев и перебоев в доступе к данным.
- Контроль входной и выходной информации:
- Необходимо реализовать механизмы контроля целостности входных данных для предотвращения ошибок и искажений при их передаче и обработке.
- Время восстановления после отказа:
- Программа должна иметь механизмы резервного копирования данных и быстрого восстановления после возможных сбоев для минимизации простоя в работе и потерь информации.

Система должна иметь стандартные и продвинутое настройки безопасности.

Для обычного пользователя обеспечить ограничение доступа к изменениям настроек системы, данная возможность есть только у администратора.

Предусматривать контроль вводимой информации и блокировку некорректных действий пользователя при работе с системой, обеспечить максимально возможную защиту от ошибок системы.

Кроме того, необходимо обеспечить возможность настройки частоты резервного копирования и восстановления – это подразумевает возможность в любой момент вернуть настройки системы и в целом базы данных к прошлому виду.

Надежное функционирование информационной системы должно быть обеспечено выполнением организационно-технических мероприятий, таких как:

- Использование лицензионного программного обеспечения;
- Организация бесперебойного питания путем использования блоков бесперебойного питания для сервера;
- Любые изменения информации в базе данных должны быть целостны и непротиворечивы.

4. Требования к составу и параметрам технических средств

- Этот раздел определяет необходимые технические характеристики для обеспечения полноценного функционирования программы.

Состав технических средств:

- Программа должна выполняться на сервере с определенными характеристиками, включая процессор, объем оперативной памяти и

дисковое пространство, достаточные для обработки и хранения данных от Zabbix.

Основные технические характеристики:

- Сервер должен поддерживать определенные сетевые протоколы и иметь достаточную пропускную способность для обмена данными с системой мониторинга.

Система должна работать на IBM совместимых персональных компьютерах.

Минимальная конфигурация:

- тип процессора Intel Core i3 или AMD Ryzen 3и выше;
- тактовая частота процессора 3.0 ГГц;
- объем оперативного запоминающего устройства около 8 Мб и выше;
- объем свободного места на жестком диске определяется объемом данных и базы данных системы, однако ожидается около 50 Мб и выше;
- платформа 32-х разрядная и выше;

Рекомендуемая конфигурация:

- тип процессора Intel Core i5 или AMD Ryzen 5;
- тактовая частота процессора 4.0 ГГц;
- объем оперативного запоминающего устройства около 128 Мб;
- объем свободного места на жестком диске около 60 Мб;
- платформа 32-х, 64-х разрядная.

5. Требования к информационной и программной совместимости

- В этом разделе указываются требования к совместимости программы с другими информационными системами и программными средствами.

Информационные структуры на входе и выходе:

- Программа должна корректно взаимодействовать с данными, поступающими из Zabbix, используя специфические API и форматы данных для сохранения совместимости.

Методы решения исходного кода:

- Язык программирования и используемые программные средства должны быть выбраны с учетом требований к производительности и безопасности обработки информации.

Защита информации и программ:

- Необходимо обеспечить защиту данных во время их передачи и хранения, включая использование шифрования и управление доступом к информации.

Программное обеспечение должно быть совместимо с операционными системами macOS, Windows 7/10/11 в соответствии с оборудованием организации.

Требования к программной документации

1. Требования к документации

Документация играет ключевую роль в обеспечении понимания, использования и поддержки программного обеспечения. Все программные модули должны быть самодокументированы, а сопровождающая документация должна быть ясной, полной и доступной для всех пользователей и администраторов системы.

- Самодокументированные программные модули:

- Каждый программный модуль должен содержать комментарии, описывающие его функциональность, входные и выходные данные, а также внутренние алгоритмы и процессы.
- Комментарии должны быть написаны на понятном языке, избегая излишней техничности, чтобы любой разработчик, работающий с кодом, мог легко понять его назначение и логику.
- Структура кода должна быть организована таким образом, чтобы логика и взаимосвязи между компонентами были интуитивно понятны, а переменные и функции — наименованы в соответствии с их ролью.

2. Состав сопровождающей документации

Сопровождающая документация должна включать несколько ключевых разделов, каждый из которых предназначен для различных групп пользователей и специалистов, участвующих в эксплуатации и поддержке программы.

- - Руководство пользователя:

- Подробное описание функциональных возможностей программы и пошаговые инструкции по выполнению основных задач.
- Примеры использования программы для решения типичных задач, с пояснениями к каждому этапу.

- Руководство администратора:

- Инструкции по установке, настройке и обновлению программного обеспечения.
- Описание процедур мониторинга работы программы, включая методы диагностики и устранения неполадок.
- Пояснения по управлению правами доступа и конфигурации системы безопасности.
- Техническое руководство:
 - Детализированное описание архитектуры программы, включая схемы модулей и их взаимодействие.
 - Описание используемых алгоритмов и подходов к обработке данных.
 - Инструкции по расширению функциональности программы и внесению изменений в код.
- Документация по API:
 - Описание всех API, используемых для интеграции с системой мониторинга Zabbix и другими внешними сервисами.
 - Примеры запросов и ответов, форматы данных и возможные коды ошибок.
- Тестовая документация:
 - Описание подходов и методов тестирования программы.
 - Сценарии тестирования для различных функциональных компонентов программы.
 - Отчеты о проведенных тестах и выявленных ошибках, с указанием способов их устранения.
- История изменений (Changelog):
 - Записи обо всех изменениях в программе, включая исправленные ошибки, добавленные функции и улучшения производительности.
 - Даты выпусков новых версий и краткие описания внесенных изменений.

5 Обоснование выбора средств реализации

Для успешной реализации программной системы необходимо тщательно выбрать инструменты и технологии, которые обеспечат надежность, масштабируемость, производительность и удобство разработки. В этом разделе рассмотрим основные критерии выбора и обоснование использования конкретных средств и технологий для реализации дипломного проекта.

Критерии выбора инструментов

Совместимость с существующими системами:

- Интеграция с системой мониторинга Zabbix.
- Возможность работы в среде Linux, так как серверное ПО будет развернуто на данной ОС.

Производительность и масштабируемость:

- Способность обрабатывать большие объемы данных в реальном времени.
- Поддержка многопоточности и распределенных вычислений.

Надежность и устойчивость к сбоям:

- Минимизация времени восстановления после отказов.
- Наличие встроенных механизмов резервирования и отказоустойчивости.

Безопасность:

- Поддержка современных методов аутентификации и авторизации.
- Защита данных при передаче и хранении.

Удобство разработки и поддержки:

- Наличие обширной документации и сообщества разработчиков.
- Поддержка модульности и масштабируемости кода.

Выбор концепции и инструментов

На основании вышеуказанных критериев, рассмотрим несколько ключевых технологий и инструментов, подходящих для реализации данного проекта.

1. Язык программирования: Go

Go был выбран в качестве основного языка программирования по следующим причинам:

- Совместимость и поддержка: Go отлично интегрируется с Zabbix через API, что позволяет легко получать и обрабатывать данные.

Производительность: хотя Go не является самым производительным языком, его расширяемость и возможность использования библиотек, написанных на C и C++, позволяют достигать необходимых показателей производительности.

Обширная экосистема: существует множество библиотек и фреймворков для веб-разработки, обработки данных, и обеспечения безопасности.

2. База данных: Zabbix

Для управления данными в данном проекте используется Zabbix, который отвечает за сбор, хранение и обработку данных о состоянии технических средств:

Надежность и производительность: Zabbix обеспечивает надежное хранение и управление данными, собирая информацию с различных устройств в реальном времени.

Интеграция: Система мониторинга Zabbix легко интегрируется с внешними приложениями через API, что позволяет нашему веб-серверу получать необходимую информацию без необходимости дублирования данных.

Масштабируемость: Zabbix поддерживает работу с большим количеством устройств и высокими нагрузками, обеспечивая масштабируемость системы

3. Фронтенд-технологии: HTML, CSS, JavaScript

Для разработки интерактивного веб-интерфейса будут использоваться:

HTML и CSS: Основные технологии для создания структурированной и стилизованной веб-страницы.

JavaScript: был выбран для создания динамического пользовательского интерфейса, позволяющего быстро и эффективно обновлять данные на странице без перезагрузки.

4. Система контроля версий: Git

Git будет использоваться для управления версиями исходного кода:

Совместная работа: Поддержка параллельной разработки и разрешения конфликтов.

История изменений: Возможность отслеживания всех изменений и возврата к предыдущим версиям.

Выбор Go и СУБД GoLand в качестве основных инструментов разработки обоснован их широкими возможностями, удобством использования и поддержкой со стороны сообщества разработчиков. PostgreSQL обеспечивает надежное хранение и управление данными, а использование JavaScript на стороне клиента позволяет создавать интерактивные и отзывчивые пользовательские интерфейсы. Такой набор инструментов обеспечивает выполнение всех функциональных и нефункциональных требований к разрабатываемой системе, а также позволяет эффективно реализовать и поддерживать программный продукт в долгосрочной перспективе.

6 Проектирование программной системы

На рисунке 6.1 представлена диаграмма, описывающая структуру программы.

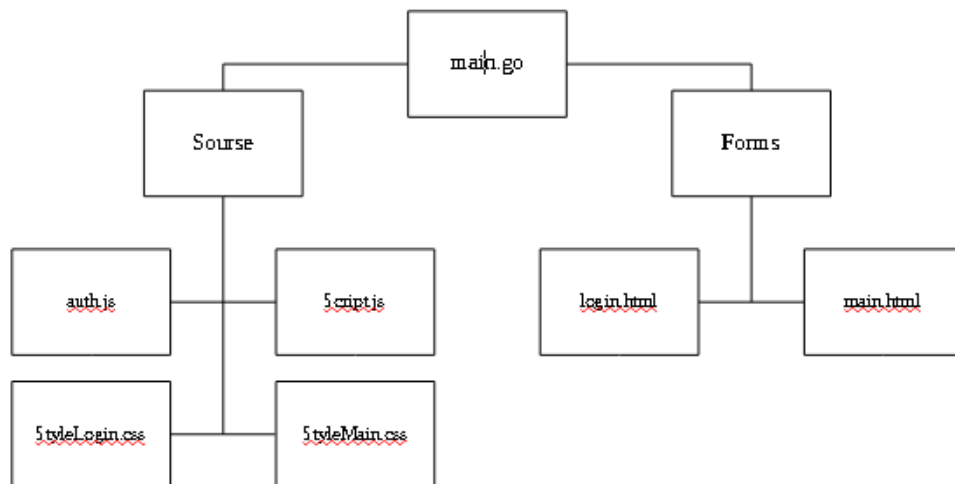


Рисунок 6.1 – Структура программы

На рисунке 6.2 представлена диаграмма, описывающая функциональную часть программы.

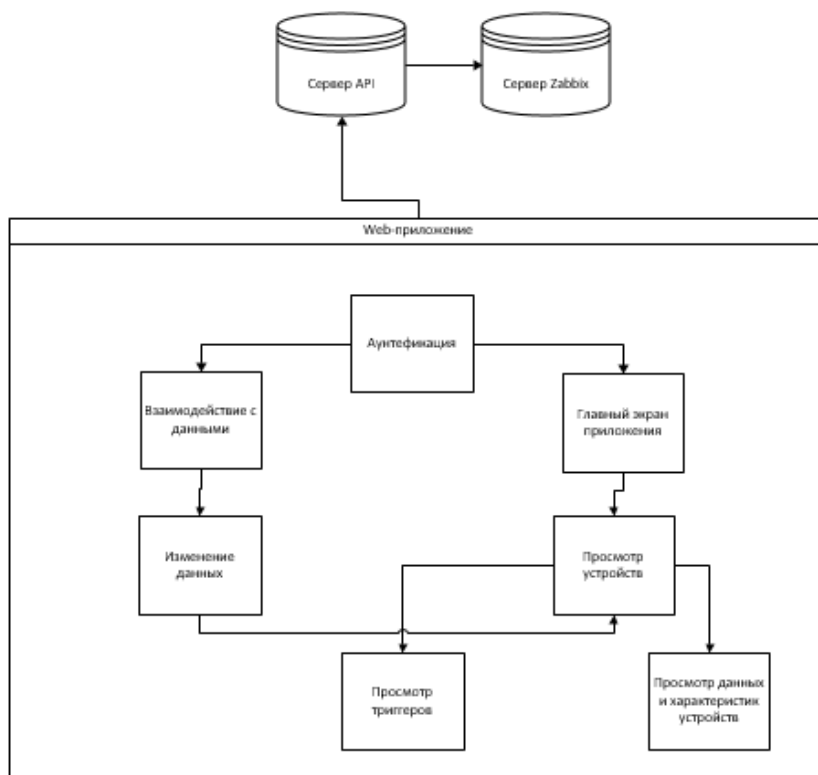
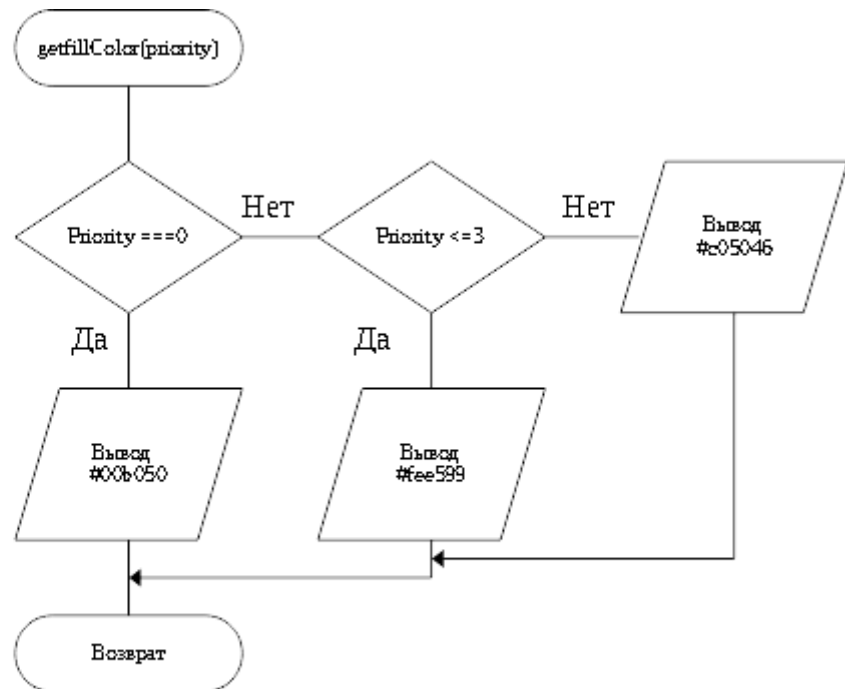


Рисунок 6.2 – Функциональная часть программы

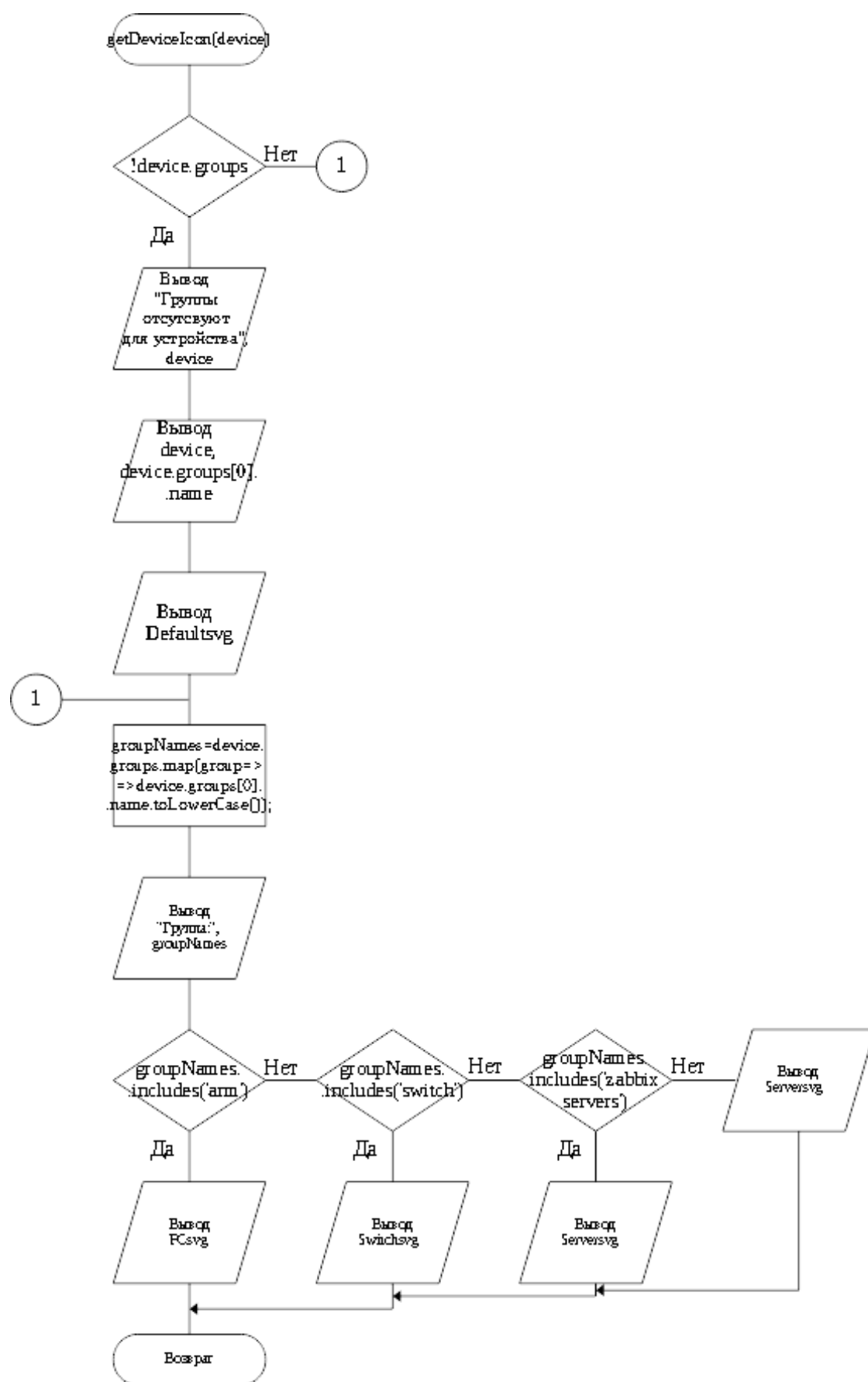
В программе структура представляет из себя форму с авторизацией, пройдя аутентификацию, вы подключаетесь к zabbix серверу, который передает данные в виде json пакетов. После успешной аутентификации мы перемещаемся на главную форму, где можно увидеть устройства в нашей сети. Данные устройства имеют свои характеристики, которые можно увидеть, нажав на панель устройства. В системе работают триггеры, которые приложение отлавливает и изменяет цвет иконки нашего устройства. Пользователь может просматривать данные об устройствах и изменять частоту обновления триггеров. Формы же соединены с js, который отвечает за функционал и с css файлами, которые отвечают за стилизацию нашего приложения.

7 Проектирование программной системы

7.1 Схема алгоритма получения цвета по приоритету



7.2 Схема алгоритма получения иконки устройства

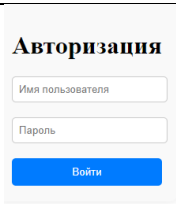
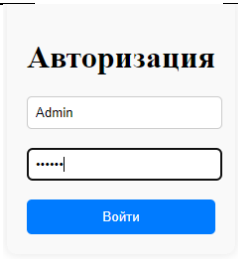
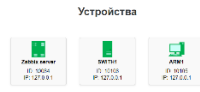



8 Проектирование программной системы

Проведем тестирование программы.

В таблице 8.1 представлен пример позитивного тест-кейса авторизации

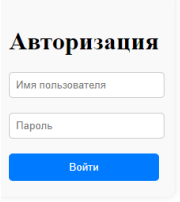
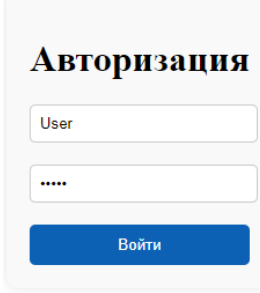
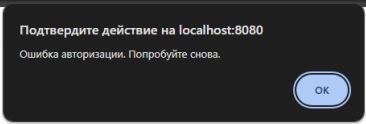
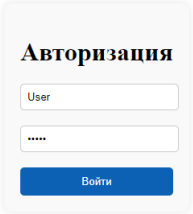

Таблица 8.1 – Пример позитивного тест-кейса авторизации

Действие	Ожидаемый результат	Результат теста
1. Запустить программу	<ul style="list-style-type: none"> – программа запущена – открыта форма авторизации 	
2. Заполнить поля формы: Логин = Admin Пароль = zabbix	<ul style="list-style-type: none"> – поля ввода заполнены – пароль скрыт системными символами 	
3. Нажать на кнопку «Войти»	<ul style="list-style-type: none"> – авторизация проходит проверку – открывается главная форма 	
4. При успешной авторизации, перейдя в наше СУБД можно увидеть, что произошла трассировка а данных	<ul style="list-style-type: none"> – Нам выдалось сообщение об успешной авторизации 	<pre>2024/06/18 13:31:09 Запуск веб-сервера на порту: 8080 2024/06/18 13:31:15 Запрос на авторизацию в Zabbix: %s(string={ "jsonrpc": "2.0", "method": "user.login", "params": { "user": "Admin", "password": "zabbix" }, "id": 1 }) 2024/06/18 13:31:15 Тело ответа от Zabbix API: {"jsonrpc":"2.0","result":{"token":"c8462b15fe70ba71c94b57d24b11392","id":1}} 2024/06/18 13:31:15 Получен ответ от Zabbix 2024/06/18 13:31:15 Успешная авторизация в Zabbix. Токен: c8462b15fe70ba71c94b57d24b11392 2024/06/18 13:31:15 Токен: c8462b15fe70ba71c94b57d24b11392 2024/06/18 13:31:15 Запрос к Zabbix API: %s(MISSING){ "jsonrpc": "2.0", "method": "host.get", "params": { "output": ["hostid","host","name","systeminfo","inventory"], "selectInterfaces": ["interfaceid","ip"], "selectGroups": ["groupid","name"], "selectItems": ["itemid","name","key","lastvalue"] }, "auth": "c8462b15fe70ba71c94b57d24b11392", "id": 1 }) 2024/06/18 13:31:16 Тело ответа от Zabbix API: {"jsonrpc":"2.0","result":[{"hostid":"10084","host":"Zabbix server","name":"Zabbix server","key":"zabbix[process,housekeeper,avg_busy]","lastvalue":"0.0000"},{"itemid":"23259","name":"Zabbix</pre>
5. Переда на сервер zabbix можно увидеть, что наши узлы совпадают	<ul style="list-style-type: none"> – Информация о устройствах 	

Далее рассмотрим негативный тест-кейс.

В таблице 8.2 приведен пример негативного тест-кейса

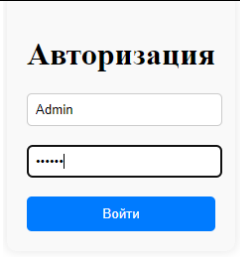
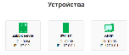
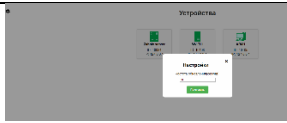
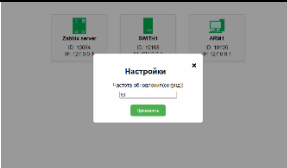
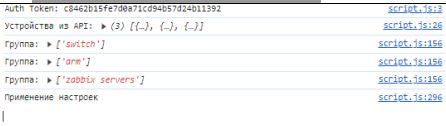
Таблица 8.2 – Пример негативного тест-кейса авторизации

Действие	Ожидаемый результат	Результат теста
1. Запустить программу	<ul style="list-style-type: none"> – программа запущена – открыта форма авторизации 	
2. Заполнить поля формы: Логин = user Пароль = none	<ul style="list-style-type: none"> – поля ввода заполнены – пароль скрыт системными символами 	
3. Нажать на кнопку «Войти»	<ul style="list-style-type: none"> – авторизация не проходит проверку – появляется сообщение с ошибкой 	 
4. Перейдя в наше СУБД, также можно посмотреть, что трассировка пакетов не прошла	<ul style="list-style-type: none"> – Zabbix сервер вернул нам ошибку 	

Проведем тестирование обработки триггера в системе.

В таблице 8.3 представлен пример позитивного тест-кейса

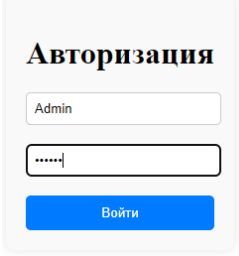
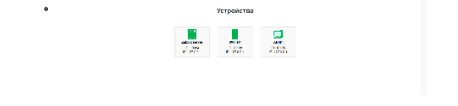
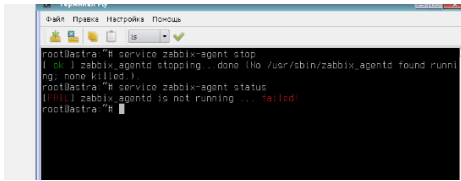
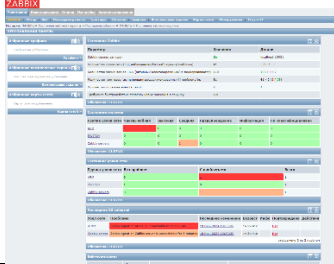
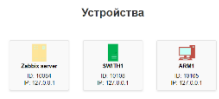

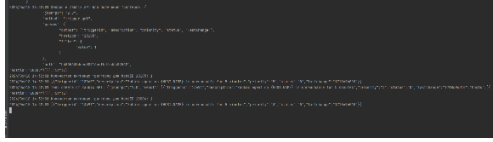
Таблица 8.3 – Пример позитивного тест-кейса обработки триггера

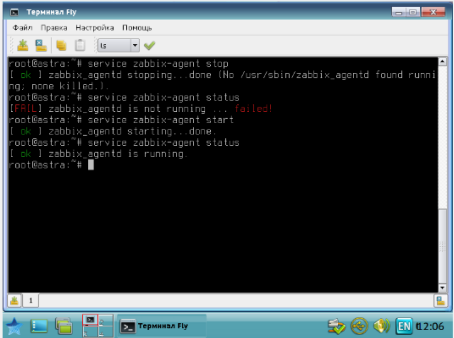
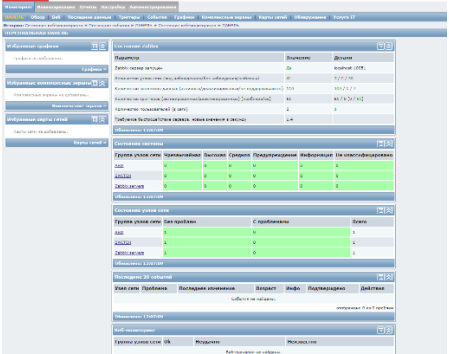



Действие	Ожидаемый результат	Результат теста
1. Запустить программу и успешно пройдем аутентификации	<ul style="list-style-type: none"> – программа запущена – открыта форма авторизации – успешная авторизация 	
2. Нажать на кнопку «Войти»	<ul style="list-style-type: none"> – открывается главная форма 	
3. Нажать на шестеренку в левом верхнем углу	<ul style="list-style-type: none"> – Выскочит изменение частоты – По умолчанию 30 секунд 	
4. Изменим на 60 секунд	<ul style="list-style-type: none"> – Вводим данные 	
5. Нажимаем на кнопку «Применить»	<ul style="list-style-type: none"> – Данные успешно применились 	

Проведем тестирование обработки триггера в системе.

В таблице 8.4 приведен пример позитивного тест-кейса регистрации

Таблица 8.4 – Пример негативного тест-кейса регистрации

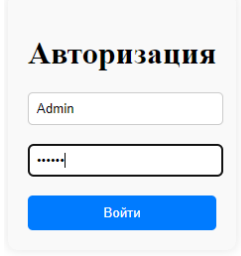
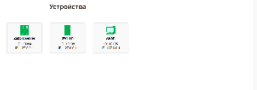
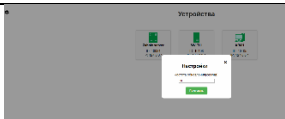
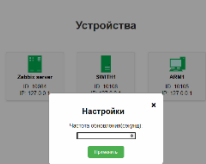
Действие	Ожидаемый результат	Результат теста
1. Запустить программу и успешно пройдем аутентификации	<ul style="list-style-type: none"> – программа запущена – открыта форма авторизации – успешная авторизация 	
2. Нажать на кнопку «Войти»	<ul style="list-style-type: none"> – открывается главная форма 	
3. Перейдем в нашу виртуалку и остановим нашего zabbix-agent	<ul style="list-style-type: none"> – Агент остановлен 	
4. Перейдем в zabbix и посмотрим на состояние устройств	<ul style="list-style-type: none"> – У устройств появились ошибки 	
5. Перейдем на наше web-приложение	<ul style="list-style-type: none"> – У устройств поменялся статус и цвет 	
6. Просмотрим детали устройств	<ul style="list-style-type: none"> – Сработал триггер 	 <div data-bbox="1374 1592 1485 1839"> <p>Детали устройства</p> <ul style="list-style-type: none"> Имя хоста: 'zabbix' Доступная память: 1.125GB Свободная память: 0.875GB Общая память: 1.950GB Общая дисковая емкость: 8.87GB Общая дисковая емкость: 14.37GB Информация о системе: Linux kernel 4.4.0-19-generic, #145ubuntu1 SMP Tue May 1 17:41:12 UTC 2016; root@zabbix Активные триггеры: 1 Активные триггеры: 1 </div>
7. Перейдем наше СУБД и посмотрим какие данные были получены	<ul style="list-style-type: none"> – Получены 2 триггера для 2 устройств 	

<p>8. Включим наш zabbix-agent в виртуалке, прописав команду</p>	<p>– Агент включен</p>	
<p>9. Перейдем в Zabbix</p>	<p>– Статус устройств поменялся, все стало зелененьким</p>	
<p>10. Перейдем на наше web-приложение</p>	<p>– У устройств поменялся статус и цвет</p>	<p>Устройства</p> <div data-bbox="1098 992 1380 1059"> <div>  <p>Zabbix agent ID: 10104 IP: 127.0.0.1</p> </div> <div>  <p>BWIN ID: 10105 IP: 127.0.0.1</p> </div> <div>  <p>ABM1 ID: 10106 IP: 127.0.0.1</p> </div> </div>

Проведем тестирование формы изменения частоты триггеров.

В таблице 8.5 представлен негативный тест-кейс изменения частоты триггеров.

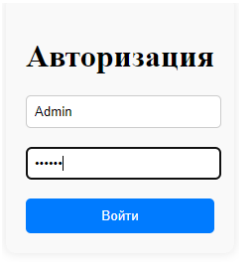
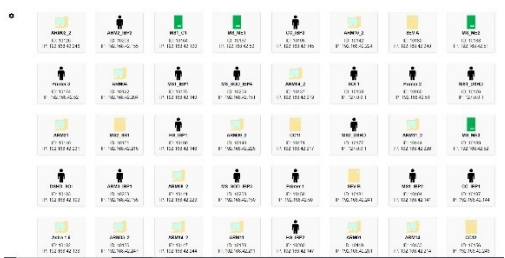


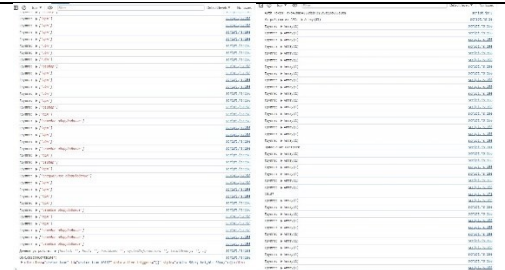
Таблица 8.5 – Пример негативного тест-кейса изменения частоты


Действие	Ожидаемый результат	Результат теста
1. Запустить программу и успешно пройдем аутентификацию	<ul style="list-style-type: none"> – программа запущена – открыта форма авторизации – успешная авторизация 	
2. Нажать на кнопку «Войти»	<ul style="list-style-type: none"> – открывается главная форма 	
3. Нажать на шестеренку в левом верхнем углу	<ul style="list-style-type: none"> – Выскочит изменение частоты – По умолчанию 30 секунд 	
4. Введем "none"	<ul style="list-style-type: none"> – Буквы не должны ввестись 	

Проведем тестирование приложения для большой сети устройств.

В таблице 8.6 представлен положительный тест-кейс тестирования

Таблица 8.6 – Пример положительный тест-кейса тестирования

Действие	Ожидаемый результат	Результат теста
1. Запустить программу и успешно пройдем аутентификацию, но под другим хостом	<ul style="list-style-type: none"> – программа запущена – открыта форма авторизации – успешная авторизация 	
2. Нажать на кнопку «Войти»	– открывается главная форма	
3. Нажать на иконку устройства	– Выдалась информация об устройстве	
4. Сохранить данные об устройстве	– Файл с данными должен загрузиться	
5. Посмотрим данные устройств на хосте	– Должны отобразить списки и группы устройств	

<p>6. Посмотрим данные устройств в нашем СУБД</p>	<p>– Должны отобразиться пакеты json, полученные от zabbix сервера</p>	
---	--	---

9 Руководство пользователя

Назначение программы

Программа предназначена для предоставления клиентам системы интерфейса для мониторинга и управления устройствами с использованием Zabbix. Клиент, авторизовавшись в системе, имеет возможность просматривать данные устройств, экспортировать их, а также редактировать и удалять информацию при наличии соответствующих прав доступа.

Условия выполнения программы

Минимальные характеристики ПК:

1. Процессор: Intel Core i3 с частотой 2.3 GHz
2. Объем оперативной памяти: 4 GB
3. Объем свободного места на жестком диске компьютера: 900 MB
4. Платформа: 32-разрядная

Рекомендуемые характеристики ПК:

1. Процессор: Intel Core i5 с частотой 3.0 GHz
2. Объем оперативной памяти: 8 GB
3. Объем свободного места на жестком диске компьютера: 3.0 GB
4. Платформа: 64-разрядная

Входные данные

Авторизация в системе происходит при помощи ввода логина и пароля. После успешной аутентификации пользователи получают доступ к различным функциям системы в зависимости от своей роли:

Пользователь:

- Просмотр данных устройств
- Экспорт данных
- Проверка наличия устройств

Администратор:

- Просмотр и редактирование данных устройств
- Удаление данных
- Обновление данных устройств

Выполнение программы

Программа запускается исполняемым файлом `main.go`. Для успешного выполнения программы необходимо убедиться, что все зависимости и необходимые библиотеки установлены.

Шаги для запуска программы:

Установка зависимостей:

Убедитесь, что у вас установлен Go и необходимые библиотеки. Для этого выполните команду:

```
sh
```

Копировать код

```
go mod tidy
```

Запуск программы: Перейдите в директорию с файлом `main.go` и выполните команду:

```
sh
```

Копировать код

```
go run main.go
```

Авторизация:

После запуска программы откройте веб-браузер и перейдите по адресу, указанному в консоли (например, `http://localhost:8080`). Введите свои учетные данные для входа в систему.

Работа с интерфейсом:

Просмотр устройств:

Перейдите в раздел "Устройства", где будет отображен список всех доступных устройств.

Экспорт данных:

Выберите устройство и нажмите кнопку "Экспорт данных" для загрузки информации.

Редактирование и удаление:

Администраторы могут редактировать и удалять данные устройств через соответствующие опции в интерфейсе.

Завершение работы

Для завершения работы программы нажмите Ctrl+C в консоли, где запущено приложение, чтобы остановить сервер. Убедитесь, что все данные сохранены и никакие важные операции не выполняются в момент завершения работы программы.

Внимание:

Перед запуском программы убедитесь, что у вас есть доступ к базе данных Zabbix и все настройки конфигурации выполнены корректно. Это необходимо для правильного функционирования системы и корректного отображения данных устройств.

Выполнение программы

Запустив программу, перейдем по локальному адресу <http://localhost:8080/login.html>. Пользователю выдаются 2 поля: поле для логина и поле для пароля. Пройдя успешно авторизацию, пользователю выдаются данные устройств, нажав на панельки, которых выдается информация об этих устройствах. Также можно задать частоту обновления устройств, нажав на колесико в левом верхнем углу. На рисунках 9.1-9.7 представлена визуальная часть.

На рисунке 9.1 представлена форма авторизации.

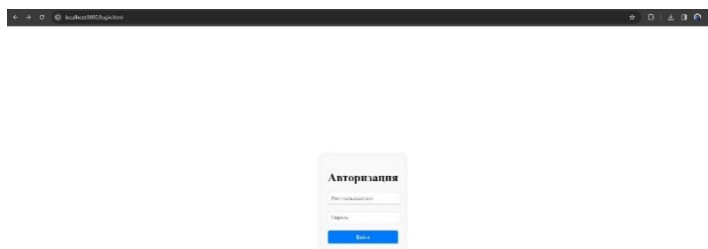


Рисунок 9.1 –Форма авторизации

Авторизуемся в приложении, введем данные.

На рисунке 9.2 представлена авторизация в веб-приложении.

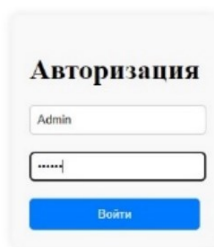


Рисунок 9.2 – Авторизация

После успешной авторизации в приложении мы перешли на главную форму и нам выдались устройства.

На рисунке 9.3 представлена главная форма с устройствами.



Рисунок 9.3 – Главная форма с устройствами

Нажмем на шестеренку, нам выдалось окно с обновлением частоты.

На рисунке 9.4 представлен результат нажатия на шестеренку.

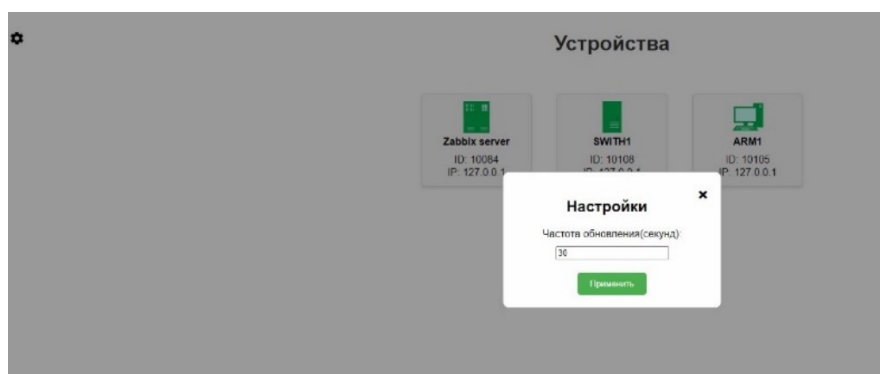


Рисунок 9.4 – Результат нажатия на шестеренку

Нажмем на панель устройств Zabbix server, нам выдались данные об устройстве.

На рисунке 9.5 представлен результат нажатия на панельку (боковая панель).

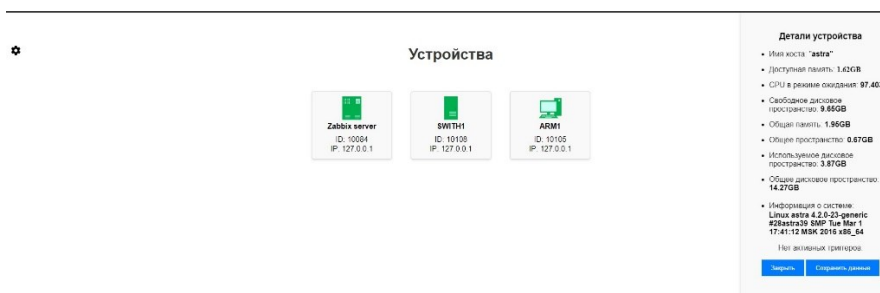


Рисунок 9.5 – Боковая панель

На боковой панели есть кнопка сохранить данные, нажмем на нее.

На рисунке 9.6 представлено выполнение сохранения данных.



Рисунок 9.6 – Сохранения данных

Откроем файл, который у нас сохранился

На рисунке 9.7 представлен просмотр итогового файла.

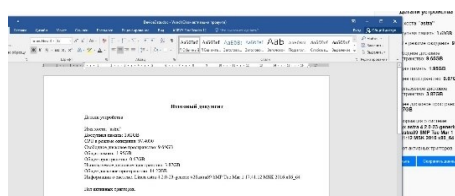


Рисунок 9.7 – Просмотр итогового файла

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана информационная система для мониторинга и управления устройствами с использованием Zabbix. В процессе реализации проекта были достигнуты следующие результаты:

Анализ предметной области:

Проведен детальный анализ текущих процессов управления устройствами и их мониторинга.

Определены основные требования к функциональным и техническим характеристикам разрабатываемой системы.

Проектирование системы:

Разработана модель предметной области с использованием современной методологии моделирования, такой как IDEF0.

Созданы диаграммы вариантов использования, отражающие взаимодействие пользователей с системой.

Реализованы основные функциональные модули системы, включая аутентификацию пользователей, просмотр и экспорт данных устройств, а также редактирование и удаление данных.

Интегрирован Zabbix в качестве основной базы данных для хранения и обработки информации об устройствах и их состоянии.

Проведены тестирования на предмет выявления и устранения ошибок.

Система внедрена в эксплуатацию, обеспечивая надежный и эффективный мониторинг устройств в режиме реального времени.

Подготовлены необходимые документационные материалы, включая руководство пользователя и техническую документацию.

Программные модули снабжены комментариями для облегчения их сопровождения и доработки в будущем.

Работа над проектом позволила достигнуть всех поставленных целей, в том числе повысить эффективность управления устройствами, обеспечить автоматизацию процессов мониторинга и обработки данных, а также улучшить качество обслуживания пользователей. Внедрение данной системы способствует снижению времени на

обработку запросов, повышению надежности функционирования устройств и предоставлению оперативной информации для принятия управленческих решений.

Таким образом, данная работа вносит значительный вклад в оптимизацию и автоматизацию процессов мониторинга и управления устройствами, что способствует повышению общей эффективности и надежности системы. В дальнейшем планируется расширение функциональности системы и ее адаптация под специфические требования различных пользователей.

СПИСОК ЛИТЕРАТУРЫ

- 1) Гражданский кодекс Российской Федерации, Закон РФ от 07.02.1992 г. №2300-І «О защите прав потребителей».
- 2) Мигель Гринберг. Flask Web Development. – М.: Юрайт, 2018.
- 3) Информационная система «Read the Docs». URL: <https://readthedocs.org>
- 4) Робсон Э. "Изучаем HTML, XHTML и CSS".
- 5) Паленов, В. Н. Базы данных / В.Н. Паленов, Д.Э. Фуфаев. -М.:Академия, 2019 год - 189 с
- 6) Лутц, Марк. Л86 Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019
- 7) "Mastering CSS: Advanced Techniques for Modern Web Design" 2019 года
- 8) Информационная система «Github» URL - <https://github.com>
- 9) Балабин, В.В., Моргунов, В.В. "Основы программирования на Go" – М.: Бином, 2020.
- 10) Саммерфилд, М. "Программирование на Go. Полное руководство", СПб: Питер, 2021.
- 11) Дэвид Херман. "Эффективный JavaScript: 68 специфических способов использовать возможности JavaScript" – М.: Диалектика, 2020.
- 12) Николас Закас. "Высокопроизводительный JavaScript" – СПб.: Питер, 2019.
- 13) Джон Дакетт. "HTML и CSS. Разработка и дизайн веб-сайтов" – М.: Вильямс, 2019.
- 14) Эрик Мейер. "CSS: The Definitive Guide" – М.: O'Reilly Media, 2020.
- 15) Олифер В.Г., Олифер Н.А. "Сети ЭВМ и телекоммуникации. Принципы, технологии, протоколы" - СПб.: БХВ-Петербург, 2021.
- 16) Грассо, Дэвид. "React и Redux: функциональная веб-разработка" – М.: Питер, 2020.
- 17) Мэннинг К., Димитри С. "Основы работы с PostgreSQL" – СПб.: Питер, 2020.
- 18) Ричардсон, Л. "Restful Web Services", O'Reilly Media, 2021.
- 19) Мартин Фаулер. "Шаблоны корпоративных приложений" – СПб.: Питер, 2020.

- 20) Джон Резиг. "JavaScript и jQuery: интерактивная веб-разработка" – М.: Диалектика, 2020.
- 21) Юдин, А. А. "Основы администрирования систем мониторинга Zabbix", СПб.: Питер, 2020.
- 22) Вандер Ньютен, Стивен. "Modern Web Development with Django" – М.: Apress, 2019.
- 23) Кривоносова Н. "Безопасность информационных систем", СПб.: Питер, 2021.
- 24) Лавринович, И. "DevOps: Внедрение и поддержка" – М.: Питер, 2019.
- 25) "Методическое пособие по использованию Zabbix", 2020. URL: <https://zabbix.com/documentation>.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программы

A.1 Листинг файла main.go

```
// ПП ПО ПМ.02 - "Осуществление интеграции программных модулей"  
// по теме: разработка программного комплекса контроля состояния  
технических средств  
// Разработал: Владимиров Иван Сергеевич  
// Группа: ТИП-62  
// Дата и номер версии: 31.05.24  
// Язык: Go  
// Краткое описание: веб-сервер для обработки информации о состоянии  
технических средств и веб-интерфейс для отображения состояния технических  
средств.  
//Задание:  
//1. Анализ и разработка диаграмм.  
//2. Разработка программного обеспечения.  
//3. Формулировка спецификации.  
//4. Заполнение и структурирование базы данных.  
//5. Создание главной формы приложения.  
//6. Реализация форм авторизации и регистрации с требованиями к  
паролям.  
//7. Создание пользовательских форм для управления данными.  
//8. Написание вычисляемых функций.  
//9. Тестирование и написание отчета.  
//10. Подготовка презентации.  
//Использованные формы:  
//login.html – форма для аутентификации пользователя;  
//main.html – основная форма для взаимодействия с устройствами.  
//Использованные функции:  
//getDeviceInfoFromZabbix(authToken, hostID string) – функция для  
выборки данных устройств с сервиса Zabbix;  
//convert(b string) – функция конвертации байтов в гигабайты;
```


`//getDevicesFromZabbix(authToken string)` - функция для выборки устройств из zabbix;

`//authenticateWithZabbix(username, password string)` – функция аутентификации пользователя в zabbix и apache2;

`//getTriggersFromZabbix(authToken, hostID string)` – функция для выборки данных триггера из zabbix сервиса;

`//fileServerHandler(w http.ResponseWriter, r *http.Request)` - функция для подгрузки файлов на наше веб-приложение.

`//Данный файл является точкой старта приложения`

`package main`

`import (`

`"encoding/json"`

`"errors"`

`"fmt"`

`"io/ioutil"`

`"log"`

`"net/http"`

`"os"`

`"strconv"`

`"strings"`

`)`

`const (`

`zabbixAPIURL = "http://192.168.42.20/zabbix/api_jsonrpc.php"`

`)`

`type Device struct {`

`ID string `json:"hostid"``

`Host string `json:"host"``

`Interfaces []struct {`

`Interfaceid string `json:"interfaceid"``

`IP string `json:"ip"``

```

} `json:"interfaces,omitempty"`
Groups []struct {
    Groupid string `json:"groupid"`
    Name    string `json:"name"`
} `json:"groups,omitempty"`
HostName    string `json:"hostName"`
SystemInformation string `json:"systemInformation"`
TotalMemory    string `json:"totalMemory"`
AvailableMemory string `json:"availableMemory"`
CPUIidleTime    string `json:"cpuIdleTime"`
TotalSwapSpace  string `json:"totalSwapSpace"`
UsedDiskSpace   string `json:"usedDiskSpace"`
TotalDiskSpace  string `json:"totalDiskSpace"`
FreeDiskSpace   string `json:"freeDiskSpace"`
}

```

```

func getDeviceInfoFromZabbix(authToken, hostID string) (Device, error) {
requestData := fmt.Sprintf(` {
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": ["key_", "name", "lastvalue"],
        "hostids": "%s",
        "filter": {
            "key_":
["status", "system.hostname", "agent.hostname", "hostid", "system.host", "system.uname",
"vm.memory.size[available]", "system.cpu.util[,idle]", "vfs.fs.size[/,free]", "vm.mem
ory.size[total]", "system.swap.size[,total]", "vfs.fs.size[/,used]", "vfs.fs.size[/,total]"]
        },
        "sortfield": "key_"
    },
    "auth": "%s",

```

```

        "id": 1
    }, hostID, authToken)

    log.Printf("Токен: %s", authToken)
    log.Printf("Запрос к Zabbix API для получения информации об устройстве: %s",
requestData)

    req, err := http.NewRequest("POST", zabbixAPIURL,
strings.NewReader(requestData))
    if err != nil {
        log.Printf("Ошибка при создании запроса: %v", err)
        return Device{}, err
    }
    req.Header.Set("Content-Type", "application/json")
    req.SetBasicAuth("pamuser", "12345678")

    client := &http.Client{}
    resp, err := client.Do(req)
    if err != nil {
        log.Printf("Ошибка при выполнении POST-запроса: %v", err)
        return Device{}, err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        log.Printf("Ошибка при чтении тела ответа: %v", err)
        return Device{}, err
    }
    log.Printf("Тело ответа от Zabbix API: %s", body)

    if resp.StatusCode != http.StatusOK {

```

```

        return Device{}, fmt.Errorf("получен неожиданный статус ответа от
сервера: %d, тело ответа: %s", resp.StatusCode, string(body))
    }

```

```

var data struct {
    Result []struct {
        ItemID   string `json:"itemid"`
        Name     string `json:"name"`
        Key      string `json:"key_"`
        LastValue string `json:"lastvalue"`
    } `json:"result"`
    Error *struct {
        Message string `json:"message"`
        Code    int    `json:"code"`
    } `json:"error"`
}

```

```

if err := json.Unmarshal(body, &data); err != nil {
    log.Printf("Ошибка декодирования ответа: %v", err)
    return Device{}, err
}

```

```

if data.Error != nil {
    log.Printf("Ошибка Zabbix API: %s (код: %d)", data.Error.Message,
data.Error.Code)
    return Device{}, errors.New(data.Error.Message)
}

```

```

deviceInfo := Device{}
for _, item := range data.Result {
    log.Printf(item.Key)
    switch item.Key {
    case "system.hostname":
        deviceInfo.HostName = item.LastValue
    }
}

```

```

case "vm.memory.size[available]":
    {
        availableMemory, err := convert(item.LastValue)
        if err != nil {
            log.Printf("Ошибка при конвертации размера
доступной памяти: %v", err)
            availableMemory = "N/A"
        }
        deviceInfo.AvailableMemory = availableMemory + "GB"
        break
    }
case "agent.hostname":
    deviceInfo.Host = item.LastValue
    break
case "hostid":
    deviceInfo.ID = item.LastValue
    break
case "system.cpu.util[,idle]":
    deviceInfo.CPUIdleTime = item.LastValue
    break
case "vfs.fs.size[/,free]":
    {
        freeDiskSpace, err := convert(item.LastValue)
        if err != nil {
            log.Printf("Ошибка при конвертации размера
доступной памяти: %v", err)
            freeDiskSpace = "N/A"
        }
        deviceInfo.FreeDiskSpace = freeDiskSpace + "GB"
        break
    }

```

```

case "vfs.fs.size[/,used]":
    {
        usedDiskSpace, err := convert(item.LastValue)
        if err != nil {
            log.Printf("Ошибка при конвертации размера
доступной памяти: %v", err)
            usedDiskSpace = "N/A"
        }
        deviceInfo.UsedDiskSpace = usedDiskSpace + "GB"
        break
    }

case "vfs.fs.size[/,total]":
    {
        totalDiskSpace, err := convert(item.LastValue)
        if err != nil {
            log.Printf("Ошибка при конвертации размера
доступной памяти: %v", err)
            totalDiskSpace = "N/A"
        }
        deviceInfo.TotalDiskSpace = totalDiskSpace + "GB"
        break
    }

case "vm.memory.size[total]":
    {
        totalMemory, err := convert(item.LastValue)
        if err != nil {
            log.Printf("Ошибка при конвертации размера
доступной памяти: %v", err)
            totalMemory = "N/A"

```

```

        }
        deviceInfo.TotalMemory = totalMemory + "GB"
        break
    }

    case "system.swap.size[,total]":
    {
        totalSwapSpace, err := convert(item.LastValue)
        if err != nil {
            log.Printf("Ошибка при конвертации размера
доступной памяти: %v", err)
            totalSwapSpace = "N/A"
        }
        deviceInfo.TotalSwapSpace = totalSwapSpace + "GB"
        break
    }

    case "system.uname":
        deviceInfo.SystemInformation = item.LastValue
        break
    }
}

log.Printf("Успешно получена информация об устройстве из Zabbix: %v",
deviceInfo)
return deviceInfo, nil
}

func convert(b string) (string, error) {
    bInt, err := strconv.ParseFloat(b, 64)
    if err != nil {
        return "", err
    }

```

```

g := bInt / (1024 * 1024 * 1024)
return fmt.Sprintf("%.2f", g), nil
}

func getDevicesFromZabbix(authToken string) ([]Device, error) {
requestData := `{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host", "name", "systeminfo", "inventory"],
        "selectInterfaces": ["interfaceid", "ip"],
        "selectGroups": ["groupid", "name"],
        "selectItems": ["itemid", "name", "key_", "lastvalue"]
    },
    "auth": "` + authToken + `",
    "id": 1
}`
log.Printf("Токен:" + authToken)
log.Printf("Запрос к Zabbix API: %s" + requestData)

req, err := http.NewRequest("POST", zabbixAPIURL,
strings.NewReader(requestData))
if err != nil {
    log.Printf("Ошибка при создании запроса: %v", err)
    return nil, err
}
req.Header.Set("Content-Type", "application/json")
req.SetBasicAuth("pamuser", "12345678")

client := &http.Client{}
resp, err := client.Do(req)
if err != nil {

```



```

        log.Printf("Ошибка при выполнении POST-запроса: %v", err)
        return nil, err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        log.Printf("Ошибка при чтении тела ответа: %v", err)
        return nil, err
    }
    log.Printf("Тело ответа от Zabbix API: %s", body)

    if resp.StatusCode != http.StatusOK {
        return nil, fmt.Errorf("получен неожиданный статус ответа от сервера: %d,
тело ответа: %s", resp.StatusCode, string(body))
    }

    var data struct {
        Result []Device `json:"result"`
        Error *struct {
            Message string `json:"message"`
            Code    int    `json:"code"`
        } `json:"error"`
    }
    if err := json.Unmarshal(body, &data); err != nil {
        log.Printf("Ошибка декодирования ответа: %v", err)
        return nil, err
    }
    if data.Error != nil {
        log.Printf("Ошибка Zabbix API: %s (код: %d)", data.Error.Message,
data.Error.Code)
        return nil, errors.New(data.Error.Message)
    }

```

```

    }
    log.Printf("Успешно получены устройства из Zabbix: %v", data.Result)
    return data.Result, nil
}

func authenticateWithZabbix(username, password string) (string, error) {
    requestData := `{
        "jsonrpc": "2.0",
        "method": "user.login",
        "params": {
            "user": "` + username + `",
            "password": "` + password + `"
        },
        "id": 1
    }`
    log.Printf("Запрос на авторизацию в Zabbix: %w", requestData)

    req, err := http.NewRequest("POST", zabbixAPIURL,
strings.NewReader(requestData))
    if err != nil {
        log.Printf("Ошибка при создании запроса: %v", err)
        return "", err
    }
    req.Header.Set("Content-Type", "application/json")
    req.SetBasicAuth("pamuser", "12345678")

    client := &http.Client{}
    resp, err := client.Do(req)
    if err != nil {
        log.Printf("Ошибка при выполнении POST-запроса: %v", err)
        return "", err
    }
}

```

```

defer resp.Body.Close()

body, err := ioutil.ReadAll(resp.Body)
if err != nil {
    log.Printf("Ошибка при чтении тела ответа: %v", err)
    return "", err
}
log.Printf("Тело ответа от Zabbix API: %s", body)
log.Printf("Получен ответ от Zabbix")
if resp.StatusCode != http.StatusOK {
    return "", fmt.Errorf("получен неожиданный статус ответа от сервера: %d,
тело ответа: %s", resp.StatusCode, string(body))
}

var responseBody map[string]interface{}
if err := json.Unmarshal(body, &responseBody); err != nil {
    log.Printf("Ошибка декодирования ответа: %v", err)
    return "", err
}
result, ok := responseBody["result"].(string)
if !ok {
    log.Printf("Ошибка: поле 'result' не является строкой")
    return "", errors.New("поле 'result' не является строкой")
}
log.Printf("Успешная авторизация в Zabbix. Токен: %s", result)
return result, nil
}

func fileServerHandler(w http.ResponseWriter, r *http.Request) {
filePath := "C:/Users/User/Desktop/Ivan/16_day/static/" + r.URL.Path
if _, err := os.Stat(filePath); os.IsNotExist(err) {
    http.Error(w, "Файл не найден", http.StatusNotFound)
}
}

```

```

        return
    }
    http.ServeFile(w, r, filePath)
}

func getTriggersFromZabbix(authToken, hostID string) ([]Trigger, error) {
    requestData := fmt.Sprintf(` {
        "jsonrpc": "2.0",
        "method": "trigger.get",
        "params": {
            "output": ["triggerid", "description", "priority", "status",
"lastchange"],
            "hostids": "%s",
            "filter": {
                "value": 1
            }
        },
        "auth": "%s",
        "id": 1
    }`, hostID, authToken)

    log.Printf("Запрос к Zabbix API для получения триггеров: %s", requestData)

    req, err := http.NewRequest("POST", zabbixAPIURL,
strings.NewReader(requestData))
    if err != nil {
        log.Printf("Ошибка при создании запроса: %v", err)
        return nil, err
    }
    req.Header.Set("Content-Type", "application/json")
    req.SetBasicAuth("pamuser", "12345678")

```

```

client := &http.Client{}
resp, err := client.Do(req)
if err != nil {
    log.Printf("Ошибка при выполнении POST-запроса: %v", err)
    return nil, err
}
defer resp.Body.Close()

body, err := ioutil.ReadAll(resp.Body)
if err != nil {
    log.Printf("Ошибка при чтении тела ответа: %v", err)
    return nil, err
}
log.Printf("Тело ответа от Zabbix API: %s", body)

if resp.StatusCode != http.StatusOK {
    return nil, fmt.Errorf("получен неожиданный статус ответа от сервера: %d,
тело ответа: %s", resp.StatusCode, string(body))
}

var data struct {
    Result []Trigger `json:"result"`
    Error *struct {
        Message string `json:"message"`
        Code    int    `json:"code"`
    } `json:"error"`
}

if err := json.Unmarshal(body, &data); err != nil {
    log.Printf("Ошибка декодирования ответа: %v", err)
    return nil, err
}

```

```

    if data.Error != nil {
        log.Printf("Ошибка Zabbix API: %s (код: %d)", data.Error.Message,
data.Error.Code)
        return nil, errors.New(data.Error.Message)
    }

    activeTriggerCount := len(data.Result)
    log.Printf("Количество активных триггеров для hostID %s: %d", hostID,
activeTriggerCount)
    //for _, trigger := range data.Result {
    //    log.Printf("Триггер: %s, Описание: %s, Приоритет: %s, Статус: %s,
Последнее изменение: %s", trigger.TriggerID, trigger.Description, trigger.Priority,
trigger.Status, trigger.LastChange)
    //}

    return data.Result, nil
}

type Trigger struct {
    TriggerID string `json:"triggerid"`
    Description string `json:"description"`
    Priority string `json:"priority"`
    Status string `json:"status"`
    LastChange string `json:"lastchange"`
}

func main() {
    http.HandleFunc("/login", func(w http.ResponseWriter, r *http.Request) {
        if r.Method != http.MethodPost {
            http.Error(w, "Метод не поддерживается",
http.StatusMethodNotAllowed)
            return

```

```

    }
    if r.Header.Get("Content-Type") != "application/json" {
        http.Error(w, "Неверный тип содержимого", http.StatusBadRequest)
        return
    }
    var creds struct {
        Username string `json:"username"`
        Password string `json:"password"`
    }
    if err := json.NewDecoder(r.Body).Decode(&creds); err != nil {
        http.Error(w, "Ошибка при декодировании JSON",
http.StatusBadRequest)
        return
    }

    zabbixAuthToken, err := authenticateWithZabbix(creds.Username,
creds.Password)
    if err != nil {
        http.Error(w, "Ошибка аутентификации в Zabbix: "+err.Error(),
http.StatusUnauthorized)
        return
    }

    http.SetCookie(w, &http.Cookie{
        Name: "zabbix_auth_token",
        Value: zabbixAuthToken,
        Path: "/",
    })

    http.Redirect(w, r, "/main.html", http.StatusSeeOther)
})
http.HandleFunc("/api/devices/", func(w http.ResponseWriter, r *http.Request) {

```

```

        cookie, err := r.Cookie("zabbix_auth_token")
        if err != nil {
            http.Error(w, "Токен аутентификации не найден",
http.StatusUnauthorized)
            return
        }
        authToken := cookie.Value

        devices, err := getDevicesFromZabbix(authToken)
        if err != nil {
            http.Error(w, "Ошибка при получении данных из Zabbix API:
"+err.Error(), http.StatusInternalServerError)
            return
        }
        w.Header().Set("Content-Type", "application/json")
        jsonData, err := json.Marshal(devices)
        if err != nil {
            http.Error(w, "Ошибка при кодировании данных в JSON:
"+err.Error(), http.StatusInternalServerError)
            return
        }
        w.Write(jsonData)
    })

```

```

    http.HandleFunc("/api/deviceinfo/", func(w http.ResponseWriter, r
*http.Request) {
        cookie, err := r.Cookie("zabbix_auth_token")
        if err != nil {
            http.Error(w, "Токен аутентификации не найден",
http.StatusUnauthorized)
            return
        }
    }

```



```

authToken := cookie.Value

hostID := r.URL.Query().Get("hostid")
if hostID == "" {
    http.Error(w, "hostID устройства не указан", http.StatusBadRequest)
    return
}

deviceInfo, err := getDeviceInfoFromZabbix(authToken, hostID)
if err != nil {
    http.Error(w, "Ошибка при получении данных устройств из Zabbix
API: "+err.Error(), http.StatusInternalServerError)
    return
}
w.Header().Set("Content-Type", "application/json")
jsonData, err := json.Marshal(deviceInfo)
if err != nil {
    http.Error(w, "Ошибка при кодировании данных в JSON:
"+err.Error(), http.StatusInternalServerError)
    return
}
w.Write(jsonData)
})

http.HandleFunc("/api/devices/triggers/", func(w http.ResponseWriter, r
*http.Request) {
    cookie, err := r.Cookie("zabbix_auth_token")
    if err != nil {
        http.Error(w, "Токен аутентификации не найден",
http.StatusUnauthorized)
        return
    }

```

```

        authToken := cookie.Value
        hostID := r.URL.Query().Get("hostid")
        if hostID == "" {
            http.Error(w, "hostID устройства не указан",
http.StatusBadRequest)
            return
        }
        triggers, err := getTriggersFromZabbix(authToken, hostID)
        if err != nil {
            http.Error(w, "Ошибка при получении триггеров из Zabbix
API: "+err.Error(), http.StatusInternalServerError)
            return
        }
        w.Header().Set("Content-Type", "application/json")
        jsonData, err := json.Marshal(triggers)
        if err != nil {
            http.Error(w, "Ошибка при кодировании данных в JSON:
"+err.Error(), http.StatusInternalServerError)
            return
        }
        w.Write(jsonData)
        log.Printf(string(jsonData))
    })

```

```

http.HandleFunc("/PC.svg", fileServerHandler)
http.HandleFunc("/Switch.svg", fileServerHandler)
http.HandleFunc("/Server.svg", fileServerHandler)
http.HandleFunc("/Default.svg", fileServerHandler)
http.HandleFunc("/GearWheel.png", fileServerHandler)
http.Handle("/",
http.FileServer(http.Dir("C:/Users/User/Desktop/Ivan/16_day/static/")))

```

```
        if _, err := os.Stat("C://Users/User/Desktop/Ivan/16_day/static/login.html");
os.IsNotExist(err) {
            log.Fatal("Файл login.html не найден в каталоге")
        }

log.Println("Запуск веб-сервера на порту: 8080")
if err := http.ListenAndServe(":8080", nil); err != nil {
    log.Fatal(err)
}
}
```

А.2 Листинг файла login.html

```
//Использованные файлы:
// styleLogin.css – стилизация нашей login формы.
// auth.js – фронтенд-скрипт для обработки формы авторизации.
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Система мониторинга - Авторизация</title>
  <link rel="stylesheet" href="styleLogin.css">
</head>
<body>
<div class="login-panel">
  <h1>Авторизация</h1>
  <form id="login-form" class="login-form" action="/login" method="post">
    <input type="text" id="username" name="username" placeholder="Имя
пользователя"><br>
    <input type="password" id="password" name="password"
placeholder="Пароль"><br>
    <button type="submit" id="login-button">Войти</button>
  </form>
</div>
<script src="auth.js"></script>
</body>
</html>
```

А.3 Листинг файла main.html

```
//Использованные файлы:
// styleMain.css – стилизация нашей main формы.
// script.js – фронтенд-скрипт для обработки главной формы.
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Устройства</title>
  <script src="script.js"></script>
  <link rel="stylesheet" href="styleMain.css">
</head>
<body>
  <div id = "settings-modal" class="settings-modal">
    <div class="contents-modal">
      <span class="close" id="close-settings">&times</span>
      <h2>Настройки</h2>
      <div class = "settings-item">
        <label for="refresh-interval">Частота обновления(секунд):</label>
        <input style="margin-top: 10px" type="number" id ="refresh-interval"
name="refresh-interval" value="30" min="1">
      </div>
      <button id="apply-settings" class="apply-bottom">Применить</button>
    </div>
  </div>
  <div style = "display: grid; flex-direction: row; flex-wrap: wrap; margin-top: 50px;
margin-bottom: 50px;" id="header">
    <div style="justify-content: left; text-align: left">
```

```

        
    </div>
    <div style="justify-content: center; text-align: center">
        <h1 style = "margin: 0px;">Устройства</h1>
    </div>
</div>
<div id="device-container" class="devices-container">
    <div id="device-list" class="device-list"></div>
</div>
<div id="sidebar" class="sidebar">
    <div id="device-details-content" class="modal-content"></div>
    <button id="close-button" class="close-button">Закрыть</button>
    <button id = "save-button" class="save-button">Сохранить данные</button>
</div>
</body>
</html>

```

А.4 Листинг файла auth.js

```
//Использованные функции:
// login – функция для передачи данных в json формате Zabbix серверу.
document.addEventListener('DOMContentLoaded', function () {
    const loginForm = document.getElementById('login-form');
    if (loginForm) {
        loginForm.addEventListener('submit', login);
    }
    function login(event) {
        event.preventDefault();
        const username = document.getElementById('username').value;
        const password = document.getElementById('password').value;
        fetch('/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ username, password })
        })
        .then(response => {
            if (response.redirected) {
                window.location.href = "main.html";
            } else {
                alert("Ошибка авторизации. Попробуйте снова.");
            }
        })
        .catch(error => {
            console.error("Ошибка:", error);
        });
    }
});
```

A.5 Листинг файла styleLogin.css

```
body{
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.login-panel{
    background-color: #f9f9f9;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}

.login-panel input {
    width: 94%;
    margin-bottom: 1px;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.login-panel button{
    width: 100%;
    padding: 10px;
    border: none;
    border-radius: 5px;
```



```
background-color: #007bff;
color: #fff;
cursor: pointer;
}

.login-panel button:hover {
    background-color: #0c61b4;
}

.login-form {
    display: flex;
    flex-direction: column;
}
```

A.6 Листинг файла styleMain.css

```
body {
    font-family: Arial, sans-serif;
    /*background-color: #f4f4f4;*/
    margin: 0;
    padding: 20px;
    text-align: center;
}
h1 {
    text-align: center;
    color: #333;
    margin-top: 50px;
    margin-bottom: 50px;
}
.device-container {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
}
.device-list {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px; /*расстояние между устр.*/
    /*max-width: 1200px;*/
}

.device-item {
    background-color: #f9f9f9;
    border: 1px solid #ddd;
```

```

border-radius: 5px;
box-shadow: 0 2px 4px rgba(0,0,0,0.1);
margin: 10px;
padding: 10px;
cursor: pointer;
transition: transform 0.2s;
text-align: center;
width: 150px;
display: flex;
flex-direction: column;
align-items: center;
}
.device-item:hover{
    transition: scale(1.05);
}
.device-item img{
    width: 50px;
    height: 50px;
    margin-bottom: 5px;
}

.sidebar {
    position: fixed;
    top: 0;
    right: -300px; /* Начальная позиция за экраном */
    width: 300px;
    height: 100%;
    background-color: #f9f9f9;
    box-shadow: -2px 0 5px rgba(0,0,0,0.1);
    transition: right 0.3s ease;
    padding: 20px;
    overflow-y: auto;

```

```

}

.sidebar.open {
    right: 0; /* Открытая позиция */
}

.close-button {
    /*background-color: #008CBA;*/
    background-color: #007bff;
    color: white;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
    text-align: center;
}

.close-button:hover {
    /*background-color: #005f6a;*/
    background-color: #0c61b4;
}

.save-button {
    /*background-color: #008CBA;*/
    background-color: #007bff;
    color: white;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
    text-align: center;
}

.save-button:hover {
    /*background-color: #005f6a;*/

```

```

    background-color: #0c61b4;
}
.settings-modal {
    display: none;
    position: fixed;
    z-index: 1000;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
    background-color: rgb(0,0,0);
    background-color: rgba(0,0,0,0.4);
}
.contents-modal{
    background-color: #fefefe;
    margin: 15% auto;
    padding: 20px;
    border: 1px solid #888;
    width: 300px;
    border-radius: 10px;
}
.close{
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
    cursor: pointer;
}
.close:hover{
    background-color: gray;
}

```

```
close:focus{
    color: black;
    text-decoration: none;
    cursor: pointer;
}

.settings-icon{
    width: 40px;
    height: 40px;
    cursor: pointer;
    position: fixed;
    /*bottom: 1020px;
    right: 10px;*/
    z-index: 1001;
}

.settings-item{
    margin: 20px;
}

.apply-bottom{
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

.apply-bottom:hover{
    background-color: #45a049;
}
```

A.7 Листинг файла script.js

```
// Используемые функции:

// document.addEventListener('DOMContentLoaded', function) – обработчик события
загрузки DOM, инициализирует программу, проверяет наличие токена и
перенаправляет пользователя на страницу входа при его отсутствии.

// getDevicesFromZabbix() – функция для получения списка устройств из Zabbix API,
обработки данных и отображения их на странице.

// fetchTriggersAndUpdateIcon(device) – функция для получения триггеров устройства
из Zabbix API, обновления иконки устройства в зависимости от приоритетов
триггеров.

// getFillColor(priority) – функция для определения цвета иконки в зависимости от
приоритета триггера.

// getDeviceIcon(device) – функция для получения пути к SVG-иконке устройства в
зависимости от его типа.

// showDeviceInfo(hostID) – функция для отображения подробной информации об
устройстве в боковой панели.

// extractContentWithoutBrackets(content) – функция для извлечения содержимого без
HTML-тегов для сохранения в документе Word.

// saveDataAsWordFile(data) – функция для сохранения данных об устройстве в файл
формата Word.

// getCookie(name) – функция для получения значения куки по имени.

// openSettings() – функция для открытия модального окна настроек.

// closeSettings() – функция для закрытия модального окна настроек.

// applySettings() – функция для применения настроек, таких как интервал обновления
данных, и их сохранения.

// Инициализация программы:

// - Получение токена аутентификации из куки.

// - Перенаправление на страницу входа при отсутствии токена.

// - Получение списка устройств из Zabbix и обновление данных с заданным
интервалом.

document.addEventListener('DOMContentLoaded', function () {

    const authToken = getCookie('zabbix_auth_token'); // Получаем
токен аутентификации из куки

    console.log('Auth Token:', authToken);
```

```

    if (!authToken) {
        // Если токен отсутствует,
        то перенаправляем пользователя на страницу входа
        window.location.href = '/login.html';
        return;
    }
    // Функция получения списка устройств из Zabbix
    function getDevicesFromZabbix() {
        fetch('/api/devices', {
            method: 'GET',
            headers: {
                'Authorization': `Bearer ${authToken}`
            }
        })
        // Установка заголовка авторизации с токеном

        .then(response => {
            if (!response.ok) {
                console.log("Данные", response);
                throw new Error('Ошибка при получении данных из Zabbix API');
            }
            return response.json();
        })
        // Парсим ответ как Json

        .then(devices => {
            console.log('Устройства из API:', devices);
            // Проверка того, что возвращает API

            const deviceList = document.getElementById('device-list');

            deviceList.innerHTML = "";
            // Очистка списка устройств перед добавлением

```



```

        devices.forEach(device => {
            перед добавлением в DOM
                const deviceItem = document.createElement('div');
                deviceItem.className = 'device-item';
                deviceItem.id = device.hostid;

                const deviceIconContainer = document.createElement('div');
                deviceIconContainer.className = 'device-icon';
                deviceIconContainer.id = `device-icon-${device.hostid}`;
                deviceIconContainer.style.width = "50px"
                deviceIconContainer.style.height = "50px"

                const deviceDetails = document.createElement('div');
                deviceDetails.className = 'device-details';
                deviceDetails.innerHTML = `
                    <h4 style="margin-top: 5px; margin-bottom: 5px;">${device.host}</h4>
                    <p style="margin-top: 10px; margin-bottom: 2px;">ID:
                    ${device.hostid}</p>
                    <p style="margin-top: 2px; margin-bottom: 2px;">IP:
                    ${device.interfaces[0].ip}</p>`;

                deviceItem.appendChild(deviceIconContainer);
                deviceItem.appendChild(deviceDetails);
                deviceList.appendChild(deviceItem);

                fetchTriggersAndUpdateIcon(device, deviceIconContainer);

                deviceItem.addEventListener('click', () => {
                    showDeviceInfo(device.hostid);
                });
            });
        });

```

```

    })
    .catch(error => {
        console.error('Ошибка загрузки устройства:', error);
    });
}

// Логика обработки и получения триггеров устройства
function fetchTriggersAndUpdateIcon(device){
    fetch(`/api/devices/triggers?hostid=${device.hostid}`)
        .then(response => {
            if (!response.ok){
                throw new Error("Ошибка при получении триггеров устройств");
            }
            return response.text()
        })
        .then(text =>{
            if(!text){
                return; // Если ответ пустой то и текст пустой
            }
            return JSON.parse(text); //Если текст не пустой парсим его (JSON)
        })
        .then(triggers => {

            if (!Array.isArray(triggers)){
                console.error("Неверный формат данных триггеров");
                return
            }

            let maxPriority = 0;
            let activeTriggers = [];

```

```

triggers.forEach(trigger => {
  if (trigger.priority > maxPriority) {
    maxPriority = trigger.priority;
  }
  if (trigger.status === "0") {
    activeTriggers.push(trigger);
  }
});

const fillColor = getFillColor(maxPriority);
const iconPath = getDeviceIcon(device);
const deviceIconContainer = document.getElementById(`device-icon-${device.hostid}`);
if(deviceIconContainer) {
  fetch(iconPath)
    .then(response => response.text())
    .then(svgText => {
      deviceIconContainer.innerHTML = svgText;
      const svgElement = deviceIconContainer.querySelector('svg');
      if(svgElement){
        svgElement.setAttribute('width','50');
        svgElement.setAttribute('height','50');
        const paths = svgElement.querySelectorAll('.bg');
        paths.forEach(path => {
          path.style.fill = fillColor;
        });

        deviceIconContainer.dataset.activeTriggers =
JSON.stringify(activeTriggers);

      } else {

```

```

        console.error("Не удалось получить документ SVG для
устройства:",device.hostid);
    }
})
.catch(error => {
    console.error("Ошибка при загрузке SVG:",error);
});
} else {
    console.error("Элемент устройства не найден для пути:",iconPath);
}
})
.catch(error => {
    console.error("Ошибка загрузки триггеров устройства:", error);
});
}
function getfillColor(priority){
    if (priority === 0) {
        return "#00b050";
    } else if (priority <= 3) {
        return "#fee599";
    } else {
        return "#c05046";
    }
}
const PCsvg = "/PC.svg"
const Switchsvg = "/Switch.svg"
const Serversvg = "/Server.svg"
const Defaultsvg = "/Default.svg"

```

// Функция получения иконки устройства в зависимости от типа

```
function getDeviceIcon(device) {  
  if(!device.groups){  
    console.log("Группы отсутствуют для устройства", device)  
    console.log("Название группы:", device.groups[0].name)  
    return Defaultsvg  
  }  

```

```
    const groupNames = device.groups.map(group =>  
device.groups[0].name.toLowerCase());  
    console.log("Группа:", groupNames)  
    if (groupNames.includes('arm')){  
      return PCsvg;  
    } else if (groupNames.includes('switch')){  
      return Switchsvg;  
    } else if (groupNames.includes('zabbix servers')){  
      return Serversvg;  
    }else return Defaultsvg;  
  }  
}
```

```
function showDeviceInfo(hostID) {  
  const sidebar = document.querySelector('.sidebar');  
  sidebar.classList.add('open');  
  console.log(hostID)
```

```
  fetch(`/api/deviceinfo?hostid=${hostID}`)  
    .then(response => {  
      if (!response.ok) {
```

```

        throw new Error("Ошибка получения данных устройства");
    }
    return response.json();
})
.then(device => {
    console.log("Данные устройств:", device)
    const modalContent = document.getElementById('device-details-content');

    if (modalContent){
        const deviceIconContainer = document.getElementById(`device-icon-${hostID}`);

        console.log("DeviceIconContainer:", deviceIconContainer);
        if (deviceIconContainer && deviceIconContainer.dataset.activeTriggers) {
            let activeTriggers =
JSON.parse(deviceIconContainer.dataset.activeTriggers);
            let triggerInfo = "";
            if (activeTriggers.length > 0) {
                triggerInfo = '<h4>Активные триггеры:</h4><ul>';
                activeTriggers.forEach(trigger => {
                    trigger.description =
trigger.description.replace('{HOST.NAME}', device.host)
                    triggerInfo += `<li style="text-align: start; margin-top: 15px; font-size: 16pt;">${trigger.description} (приоритет: ${trigger.priority})</li>`;
                });
                triggerInfo += '</ul>';
            } else {
                triggerInfo = '<p>Нет активных триггеров.</p>';
            }

            modalContent.innerHTML = `
<h3 style="text-align: center">Детали устройства</h3>

```

```

<ul>

    <li style="text-align: start; margin-top: 15px; font-size: 16pt;"> Имя
хоста: "<h style="text-align: start;font-weight:bolder; font-size: 16pt">${device.hostName
|| "N/A"}<h/>"</li>

    <li style="text-align: start; margin-top: 15px; font-size: 16pt;">
Доступная память: <h style="text-align: start;font-weight:bolder; font-family: 'Arial, sans-
serif'; font-size: 16pt" >${device.availableMemory || "N/A"}<h/></li>

    <li style="text-align: start; margin-top: 15px; font-size: 16pt;"> CPU в
режиме ожидания: <h style="text-align: start;font-weight:bolder; font-size: 16pt"
>${device.cpuIdleTime || "N/A"}<h/></li>

    <li style="text-align: start; margin-top: 15px; font-size: 16pt;">
Свободное дисковое пространство: <h style="text-align: start;font-weight:bolder; font-
size: 16pt" >${device.freeDiskSpace || "N/A"}<h/></li>

    <li style="text-align: start; margin-top: 15px; font-size: 16pt;"> Общая
память: <h style="text-align: start;font-weight:bolder; font-size: 16pt"
>${device.totalMemory || "N/A"}<h/></li>

    <li style="text-align: start; margin-top: 15px; font-size: 16pt;"> Общее
пространство: <h style="text-align: start;font-weight:bolder; font-size: 16pt"
>${device.totalSwapSpace || "N/A"}<h/></li>

    <li style="text-align: start; margin-top: 15px; font-size: 16pt;">
Используемое дисковое пространство: <h style="text-align: start;font-weight:bolder;
font-size: 16pt" >${device.usedDiskSpace || "N/A"}<h/></li>

    <li style="text-align: start; margin-top: 15px; font-size: 16pt;"> Общее
дисковое пространство: <h style="text-align: start;font-weight:bolder; font-size: 16pt"
>${device.totalDiskSpace || "N/A"}<h/></li>

    <li style="text-align: start; margin-top: 20px; font-size: 16pt;">
Информация о системе: <br><p style="font-weight:bolder; margin-top: 2px; font-size:
16pt" >${device.systemInformation || "N/A"}<p/></li>

</ul>

    ${triggerInfo}`;
}

else {

    console.error("Не удалось найти SVG элемент для устройства или
данные триггеров не доступны:",hostID)

}

} else {

```

```

        console.error("Элемент с таким ID не найден");
    }
})
.catch(error => {
    console.error("Ошибка загрузки данных устройства:", error);
});
};

const saveButton = document.getElementById('save-button');
saveButton.addEventListener('click',()=>{
    const modalContent = document.getElementById('device-details-content');
    if(modalContent){
        const data = extractContentWithoutBrackets(modalContent.innerHTML);
        saveDataAsWordFile(data);
    }
    else {
        alert('Нет данных для сохранения');
    }
});

```

```

function extractContentWithoutBrackets(content){
    let text =
content.replace(/<li[^>]*>|<\li>|<h3[^>]*>|<\h3>|<p[^>]*>|<\p>|[\{\}\[\]]|<h[^>]*>|<\h>|<
ul[^>]*>|<\ul>|<br[^>]*>|<h4[^>]*>|<\h4>|/g,"");
    text = text.trim();
    text = text.split('\n').map(line => line.trim()).join('\n');
    return text
}

function saveDataAsWordFile(data){
    const header = `
<!DOCTYPE html>

```



```

<html xmlns="http://www.w3.org/1999/xhtml"
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Итоговый документ</title>
  <div style="text-align: center; font-weight: bold; font-size: 16px; font-family: 'Times
New Roman';">
    Итоговый документ
  </div>
  <div style="height: 24pt;"></div>
  <div style = "font-size: 13px; font-family: 'Times New Roman';">
    ${data.replace(/\n/g,'<br>')}
  </div>
`;
const blob = new Blob([header], {type: 'application/msword'});
const url = URL.createObjectURL(blob);
const a = document.createElement('a');
a.href = url;
a.download = 'DeviceData.doc';
document.body.appendChild(a);
a.click();
document.body.removeChild(a);
}

// Обработчик закрытия боковой панели
const closeSidebarButton = document.querySelector('.close-button');
closeSidebarButton.addEventListener('click', () => {
  const sidebar = document.querySelector('.sidebar')
  sidebar.classList.remove('open');
});

```

```

// Функция получения значения куки
function getCookie(name) {
    const value = `; ${document.cookie}`;
    const parts = value.split('; ${name}=');
    if (parts.length === 2) return parts.pop().split(';').shift();
}

let refreshInterval = 30;
let refreshIntervalId;

const settingsIcon = document.getElementById('settings-icon')
const closeSettingsBtn = document.getElementById('close-settings')
const applySettingsBtn = document.getElementById('apply-settings')

if (settingsIcon && closeSettingsBtn && applySettingsBtn){
    settingsIcon.addEventListener("click", openSettings);
    closeSettingsBtn.addEventListener("click", closeSettings)
    applySettingsBtn.addEventListener("click", applySettings)
}
else {
    console.error("Элементы настройки не найдены")
}

function openSettings(){
    document.getElementById("settings-modal").style.display = "block"
}

function closeSettings(){
    document.getElementById("settings-modal").style.display = "none"
}

```

```

}

function applySettings(){
    console.log("Применение настроек")
    const refreshInput = document.getElementById("refresh-interval");
    const newInterval = parseInt(refreshInput.value, 10);

    if (!isNaN(newInterval) && newInterval > 0) {
        refreshInterval = newInterval;
        clearInterval(refreshIntervalId);
        refreshIntervalId = setInterval(getDevicesFromZabbix, refreshInterval * 1000);
    }

    closeSettings();
}

refreshIntervalId = setInterval(getDevicesFromZabbix, refreshInterval * 1000);
getDevicesFromZabbix();
});

```

ПРИЛОЖЕНИЕ Б
(обязательное)
Результаты выполнения программы

93

Дневник

прохождения учебной практики ПП.02 ПМ.02 Осуществление интеграции программных модулей


по профессиональному модулю ПМ.02 Осуществление интеграции программных модулей в объеме 144 часов

студента(ки) группы Владимирова И.С. --- ПИИ-62

Код, специальность 09.02.07 Информационные системы и программирование. Квалификация – Программист.

Дата	Тема занятия	Объем выполненной работы	Отметка о выполнении (выполнено/не выполнено)
28.05.24	Вводный инструктаж. Первичное ознакомление с организацией. Ознакомление с должностными инструкциями, рабочим местом и оборудованием. Ознакомление с техникой безопасности на рабочем месте. Изучение внутреннего трудового распорядка.	6	Выполнено
29.05.24	Организация рабочего места и обсуждение задачи на практику (техническое задание).	6	Выполнено
30.05.24	Обсуждение проектирования приложения по теме предприятия. Анализ предметной области.	6	Выполнено
31.05.24 - 3.06.24	Написание кода для работы с информацией. Реализация подключения и передачи пакетов Zabbix.	24	Выполнено
4.06.24 - 10.06.24	Конструирование прототипа программы. Разработка интерфейса и функциональной части программы.	36	Выполнено
11.06.24 - 15.06.24	Интегрирование программных модулей. Подключение сервиса Zabbix к программному модулю. Отладка приложения.	30	Выполнено
17.06.24 - 21.06.24	Сдача приложения, оценка куратора. Отладка и тестирование приложения.	30	Выполнено
22.06.24 - 24.06.24	Заполнение отчёта и сдача документации по пройденной практике на предприятии.	6	Выполнено

Руководитель практики от
ПРЕДПРИЯТИЯ

 Полкин ПИИ

(подпись, расшифровка подписи)

«24» 06 20 24 г.

по производственной практике
студента (ки) Московского техникума космического приборостроения МГТУ им. Н.Э. Баумана

Владимирова Елена Сергеевна
(Ф.И.О. студента)

Группа **ТИП- 62**

Специальность **09.02.07 Информационные системы и программирование** Квалификация -
Программист

(код, наименование специальности)

прошел (ла) производственную практику **ПП 02 ПМ.02 Осуществление интеграции программных модулей**

(наименование практики)

по профессиональному модулю **ПМ.02 Осуществление интеграции программных модулей** в
объеме 144 часов

(наименование профессионального модуля)

с «28» мая 2024 года по «24» июня 2024 года

на предприятии (организации) АО Корпорация "Навигатор", Велозаводская ул., 5,
Москва

(юридический адрес предприятия (организации))

Виды и качество работ в период производственной практики

Виды работ, выполненные студентом во время практики, согласно программы производственной практики	Результат (по 5-ти бальной шкале)
Осуществление интеграции программных модулей и соответствующими ему компетенциями	5 (на 5)

В ходе производственной практики студентом освоены следующие профессиональные компетенции

Код и название профессиональной компетенции	Результат освоения (освоена/не освоена)
ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.	освоена
ПК 2.2. Выполнять интеграцию модулей в программное обеспечение	освоена
ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств	освоена
ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.	освоена
ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.	освоена

Характеристика профессиональной деятельности обучающегося во время производственной

практики: Во время производственной практики я выполняла следующие задачи: анализировала требования к программным модулям, участвовала в разработке программного обеспечения, осуществляла интеграцию модулей, проводила отладку программного модуля, разрабатывала тестовые наборы и тестовые сценарии, проводила инспектирование компонент программного обеспечения.

Рекомендуемая оценка по практике 5 (на 5)

Руководитель практики от предприятия (организации)

нач. сектора
Должность

Подпись

М.П.

Роман ГИ

Ф.И.О. руководителя практики

Итоговая оценка по практике отлично

Руководитель практики от образовательного учреждения

преподаватель
Должность

Подпись



Гришкова О.В.
Ф.И.О. руководителя практики

Назаров
19.06.2024