КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА ВЕБ-ИНТЕРФЕЙСА УПРАВЛЕНИЯ УМНЫМ ДОМОМ

Кузьменко И. А., Мозговенко А.А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: ya@amozgovenko.ru

Существующие решения для управления умным домом требуют создания эффективной и масштабируемой архитектуры, обеспечивающей безопасное взаимодействие между компонентами системы в режиме реального времени.

Материалы и методы исследования

Современные исследования в области автоматизации жилья акцентируют внимание на важности интеграции различных устройств в единую систему. Особое значение придается обеспечению безопасности передачи данных и возможности масштабирования системы.

Целью работы является разработка и анализ клиент-серверной архитектуры веб-интерфейса управления умным домом, обеспечивающей:

– Надежное взаимодействие компонентов системы

– Безопасную передачу данных

– Возможность масштабирования

– Работу в режиме реального времени

Результаты исследования и их обсуждение

Серверная часть реализована на языке программирования Node.js и современного фреймворка Express. Для правильной настройки сервера будут использоваться следующие протоколы:

– протокол HTTPS

– протокол MQTT

– WEBSockets

Протокол HTTPS распределяется на две части:

– HTTP(HyperText Transfer Protocol) – относится к списку не защищенных протоколов

– HTTPS(HyperText Transfer Protocol Secure) – протокол является расширением протокола HTTP с поддержкой шифрования для повышения уровня безопасности

Согласно схеме взаимодействия сервера с приложением (рис. 1) все участники одной сети контактируют между собой с помощью сервера. Чтобы обеспечить беспрерывную передачу данных между сервером и веб-приложением, используется подключение по WebSocket-у, и по протоколу подключения MQTT отправляются данные с умных устройств на сервер.

Серверная часть разработана по стандарту CRUD:

1. Create – создание данных
2. Read – считывание данных
3. Update – обновление данных
4. Delete – удаление данных

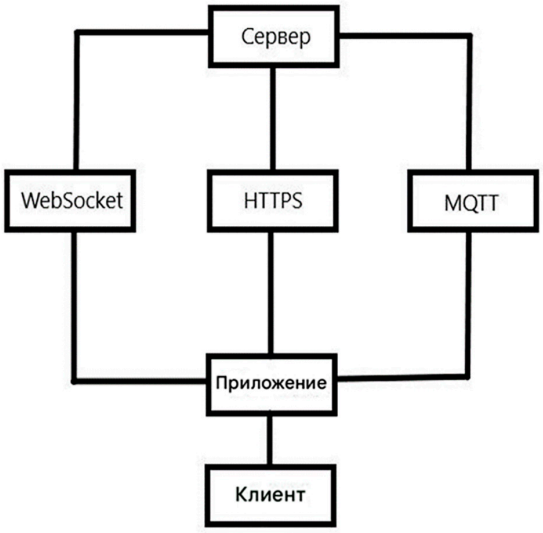

Рис. 1. Схема взаимодействия сервера с приложением

Хранилищем для хранения серверных данных выступает MongoDB. MongoDB представляет собой документно ориентированную базу данных и относится к типу NoSQL баз данных. MongoDB не является реляционной базой данных и использует коллекции и документы. Основной единицей данных служат документы, состоящие из пар ключ-значений. В коллекциях содержатся наборы функций и документов, являющихся эквивалентом таблиц реляционной базы данных.

MongoDB широко используется в сотрудничестве с AWS, Azure и другими источниками данных для разработки и функционирования приложений. Позволяя хранить и запрашивать большие объемы данных, он предлагает следующие надежные функции:

– Лучшее выполнение запросов с надлежащими функциями индексации и обработки.

– Аналитика в реальном времени и оптимизированная обработка данных с использованием специальных запросов.

– Улучшена доступность и гибкость данных благодаря надежным функциям репликации.

– Совместное использование данных позволяет разделять большие блоки данных для распределенного и более быстрого процесса выполнения запросов.

MongoDB хранит данные в формате JSON с парами ключей и значений для каждой сущности (рис. 2), в то время как базы данных SQL хранят данные в виде записи в строке таблицы (рис. 3).

MongoDB является лучшей базой для горизонтального и простого масштабирования. Гибкая база данных, которую можно постоянно совершенствовать, добавлять больше серверов, расширять хранилища и настраивать их и имеет

такие преимущества, как высокая производительность, простота развертывания, простота использования и удобное хранение данных. Данные группируются и хранятся в наборе данных. Каждый набор данных представляет собой набор, каждая база данных содержит несколько наборов.

На рисунке 4 наглядно представлена клиентская часть, на которой реализована форма входа и форма регистрации для новых пользователей.

Веб-приложение разработано на языке программирования TypeScript с использованием одной из самых популярных библиотек React. TypeScript является тем же языком программирования JavaScript, но имеет более расширенные возможности, чем JavaScript и более типизированную структуру кода.

Для обеспечения соединения сервера с приложением используются FETCH запросы и для получения данных в реальном времени используется WebSocket.

```
{
  "orderID": 12345,
  "shopperName": "Ivan Ivanov",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Super product",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Wonderful product",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

Рис. 2. Формат данных JSON

Name	age	contact-mobile	home-address
Perry	20	9273723723	perry street

Рис. 3. Формат данных SQL

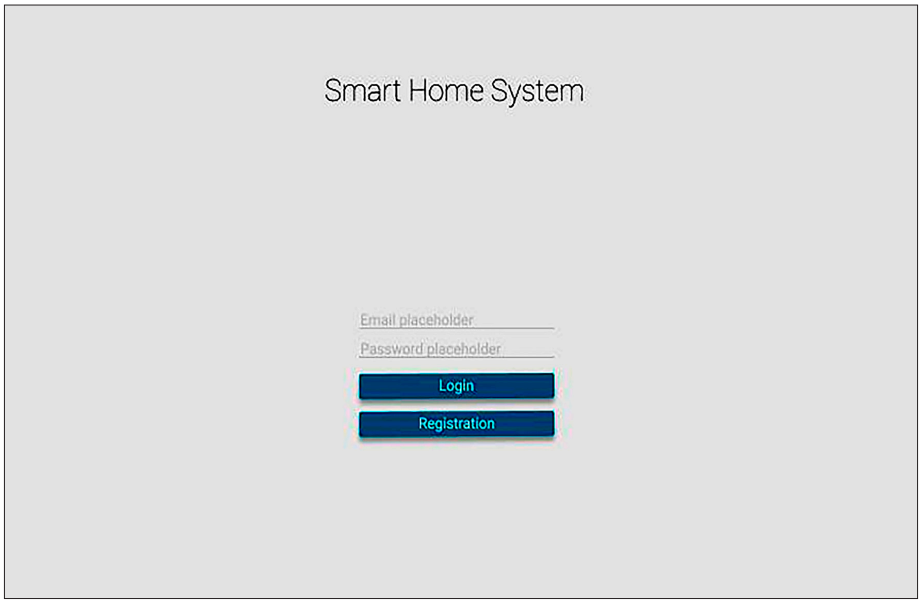

Рис. 4. Smart Home Systems форма входа

FETCH запросы делятся на методы:

– POST – для отправки данных на сервер

– GET – для получения данных с сервера

– PATCH – для обновления данных на сервере

– DELETE – для удаления данных с сервера

Чтобы открыть новый веб-сокет соединения, сначала нужно сделать запрос на подключение указав ссылку со специальным протоколом ws, равно как и HTTP, веб-сокет может быть связан с зашифрованным протоколом wss (рис. 5). Следует отметить, что лучше использовать зашифрованные протоколы подключения к серверам, так как именно такие протоколы гарантируют безопасный транспортный уровень и шифрует все данные от отправителя и расшифровывает их на стороне защитника от злоумышленников.

После инициализации нового сокет подключения следует разработать общение между сервером и приложением (рис. 6).

Переменная socket представляет собой объект с четырьмя функциями:

– onopen – функция, устанавливающая подключение между сервером и приложением

– onmessage – функция, которая после успешно установленного соединения передает данные с сервера в приложение.

– onclose – функция, закрывающая подключение между сервером и приложением

В случае разорванного соединения или ошибки сервера

– onerror – функция, выводящая все ошибки сервера

Для хранения серверных данных используется state-management библиотека Redux с использованием библиотеки redux-thunk для обработки асинхронных запросов. Библиотека Redux включает в себя основные модули для разработки:

– Slice(Reducer)

– Actions

– Selector’s

Самым главным модулем является Slice, в этом модуле разрабатываются редьюсеры и сохраняется логика обработки и записи данных в перемене через actions. Вслед за Slice по приоритетности следуют Actions, где хранятся асинхронные функции отправляющие и получающие данные с сервера, сами actions являются очень гибкими, в них можно описывать не только логику обработки серверных данных, но и управлять состоянием компонента на «расстоянии», передавать локальные данные, уход. Selector’s представляют собой функции в которых описаны ссылки на переменные в Slice, иными словами, через Selectors извлекаются серверные данные.

```
const socket = new WebSocket('wss://javascript.info');
```

Рис. 5. Веб-сокет с зашифрованным протоколом

```
socket.onopen = function (event) {
    console.log('[open] Connected', event);
    console.log('Sending data tto the server');
};

socket.onmessage = function (event) {
    console.log(`[message] Getting data from the server: ${event.data}`);
};

socket.onclose = function (event) {
    if (event.wasClean) {
        console.log(`[close] Connection closed, code=${event.code} reason=${event.reason}`);
    } else {
        console.log('[close] Connection dropped');
    }
};

socket.onerror = function (error) {
    console.log(`[error] ${error.message}`);
};
```

Рис. 6. Общение между сервером и приложением

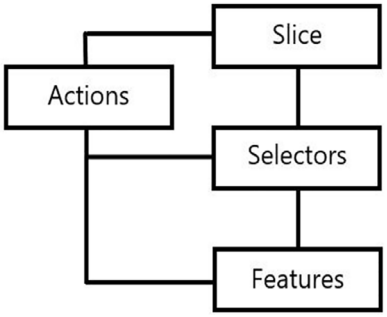

Рис. 7. Схема взаимодействия Redux с Features

На рисунке 7 представлена модель взаимодействия Redux с компонентами, где хранится основная логика приложения – такие компоненты называются Features (основные компоненты, в которых сохраняется логика обработки запросов).

Заключение

Разработанная архитектура демонстрирует эффективное взаимодействие между компонентами системы умного дома. Использование современных технологий обеспечивает:

– Надежную защиту данных через HTTPS

– Оперативную передачу информации посредством WebSocket

– Гибкое хранение данных в MongoDB

– Удобный интерфейс управления на базе React

Система готова к масштабированию и может быть адаптирована под различные задачи автоматизации жилья.

Список литературы

1. Белов А. В. Практическая энциклопедия Arduino. М.: Наука и техника. ДМК Пресс, 2018. 272 с.

2. Блум Д. Изучаем Arduino Инструменты и методы технического волшебства: учебное пособие. М: БХВ-Петербург, 2016. 336 с.

3. Геддес М. 25 крутых проектов с Arduino. М.: Эксмо, 2016.

4. Иго Т. Arduino, датчики и сети для связи устройств. М.: БХВ-Петербург, 2017. 544 с.

5. Володин В. Д., Шаронов А. А., Полевщиков И. С. Средства разработки и отладки программного обеспечения отечественных микропроцессорных устройств (часть 2) // Science Time. 2016. № 1(25). С. 91-94. EDN: VLIUFJ.

АНАЛИЗ СТАНДАРТОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ И РАЗРАБОТКА ОПЕРАЦИОННОЙ МОДЕЛИ ДЛЯ ПРЕДПРИЯТИЙ КРИТИЧЕСКОЙ ИНФРАСТРУКТУРЫ

Мельников А. В., Мозговенко А. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: ya@amozgovenko.ru

Задачи исследования:

1. Провести сравнительный анализ международных и национальных стандартов ИБ, применимых к КИ.

2. Выявить типовые угрозы и сценарии атак на объекты КИ (с учётом специфики энергетики, транспорта, ТЭК, связи и др.).

3. Определить ключевые требования регуляторов (ФСТЭК, ФСБ, Минцифры) к защите КИ.

Материалы и методы исследования

Современные исследования в области ИБ критической инфраструктуры фокусируются на следующих аспектах:

1. Регуляторная среда. Активно анализируются требования ФЗ-187, приказов ФСТЭК № 239 и № 31, а также международные стандарты (ISO/IEC 27001, NIST CSF, IEC 62443 для промышленных систем). Отмечается тенденция к гармонизации российских и зарубежных норм.

2. Киберугрозы для КИ. Публикации 2023–2025 гг. выделяют рост целевых атак (APT) на SCADA/ICS, использование вредоносного ПО типа TRITON/TRISIS, а также угрозы со стороны инсайдеров. Особое внимание уделяется уязвимостям устаревших промышленных протоколов (Modbus, DNP3).

3. Технологии защиты. Исследуются решения для:

− поведенческого анализа трафика (NTA);

− защиты конечных точек в промышленных сетях (EDR для ICS);

− автоматизации реагирования (SOAR);

− киберполигонов для тестирования устойчивости КИ.

4. Управление рисками. Развиваются методики количественной оценки рисков для КИ с учётом каскадных эффектов (например, отключение энергоснабжения ведёт к остановке транспорта).

Цель исследования – проанализировать операционную модель информационной безопасности для предприятий критической инфраструктуры, обеспечивающую соответствие регуляторным требованиям и устойчивую защиту от актуальных киберугроз.

Результаты исследования и их обсуждение

Для внедрения комплексной системы защиты информации и системы управления информационной безопасностью разработан ряд государственных и международных стандартов.

Обычно используют следующие стандарты для анализа, внедрения и постоянного мониторинга за СУИБ:

1. ГОСТ Р ИСО/МЭК 27001-2021 – российский аналог международного стандарта ISO/IEC 27001:2013. Устанавливает требования к созданию, внедрению, поддержанию и постоянному улучшению системы менеджмента информационной безопасности (СМИБ). Включает оценку и обработку рисков, выбор мер защиты, мониторинг эффективности и непрерывное совершенствование системы.