УДК 004.771

ПОДХОДЫ К РАЗРАБОТКЕ ИНТЕРФЕЙСА АССИСТЕНТА ДЛЯ УПРАВЛЕНИЯ УМНЫМ ДОМОМ

К.В. МАРКЕВИЧ
(Представлено: канд. техн. наук, доц. А. Ф. ОСЬКИН)

В статье представлен практический способ создания интерфейса ассистента для управления умным домом. Проведён анализ технологий, наиболее подходящих, для разработки данной системы. Задача, реализация прототипа интерфейса с использованием веб-компонентов.

Введение. Автоматизация здания – это автоматическое централизованное управление системами отопления, вентиляции и кондиционирования, освещением, контролем доступа, системами безопасности и другими взаимосвязанными системами через системы автоматизации зданий (BMS) или систему централизованного контроля и надзора технических установок в здании (BAS). Целями автоматизации здания являются повышение комфорта жильцов, эффективная работа систем здания, снижение энергопотребления, снижение эксплуатационных расходов, повышение безопасности, документация производительности, удаленный доступ / управление / эксплуатация, а также улучшение жизненного цикла оборудования и связанных с ним коммунальных услуг.

Автоматизация зданий является примером распределенной системы управления – компьютерной сети электронных устройств, предназначенных для мониторинга и управления системами в здании.

Здание, управляемое BAS, часто называют интеллектуальным зданием «умным зданием» или (если это жилой дом) «умным домом» [1].

Выбор технологий разработки интерфейса.

Выбор средств решения поставленной задачи исходил из того, каким программное и аппаратное обеспечение необходимо для разработки. Здесь выбор был сделан в пользу веб-приложения.

В качестве среды разработки для реализации проекта было выбрано Visual Studio Code.

Основные возможности и преимущества программы:

- Visual Studio Code поддерживает работу с TypeScript, JavaScript, Node.js и Mono.
- Имеются встроенные отладчик и командная строка.
- Поддержка практически всех языков программирования.
- Наличие встроенной библиотеки элементов кода.
- Автозавершение при вводе кода.
- Добавление в библиотеку собственных сниппетов.
- Подсветка синтаксиса.
- Одновременная работы с несколькими проектами.
- Поддержка многооконного и двухпанельного режимов.
- Расширение функционала с помощью плагинов.
- Интеграция с Visual Studio Team Services, GitHub и GIT.
- Наличие встроенных средств для тестирования, сборки, упаковки и развертывания приложений.
- Публикация созданных программных продуктов в Microsoft Azure (через посредство Visual Studio Team Services).
- Интегрированная система подсказок.
- Командная работа над проектами.
- Широкий набор настроек и кроссплатформенность.

В качестве языка программирования для написания приложения был выбран Python. Python, как и любой другой язык программирования, имеет свои отличительные особенности. Итак, можно выделить следующие:

Кроссплатформенность. Python – это интерпретируемый язык, его интерпретаторы существуют для многих платформ. Поэтому с запуском его на любой ОС не должно возникнуть проблем.

С Python доступно огромное количество сервисов, сред разработки, и фреймворков. Легко можно найти подходящий продукт для работы.

Возможность подключить библиотеки, написанные на C. Это позволяет повысить эффективность, улучшить быстродействие.

Наличие самых разных источников информации о Python. Не составит труда найти ответ на любой возникший вопрос, так существует много бесплатной литературы, обучающих видео-пособий, готовых исходников и шаблонов для работы в открытом доступе [2].

Проектирование интерфейса.

Интерфейс можно разделить на 4 части:

- Bootstrap. Это очень крошечный скрипт, который загружается на страницу в первую очередь. Он отвечает за проверку учетных данных для аутентификации и настройку соединения веб-сокета с серверной

частью. Скрипт позволяет нам начать загрузку данных, одновременно загружая остальную часть пользовательского интерфейса.

- Оболочка приложения. Это все, что требуется для рендеринга боковой панели и обработки маршрутизации.
- Панели. Каждая страница представляет собой панель. Компоненты могут регистрировать дополнительные панели, которые будут отображаться пользователю. Примерами панелей являются

«состояния», «карта», «бортовой журнал» и «история».

- Диалоги. Определенная информация и ввод данных предоставляются пользователям в потоках. Диалоги можно запускать на любой странице. Наиболее распространенным является диалоговое окно с дополнительной информацией о сущности, которое позволяет пользователям копаться в состоянии сущности, истории и настройках.

Интерфейс использует API Websocket и Rest API для взаимодействия ассистентом.

Данные доступны в виде hass свойства, которое передается каждому компоненту. Hass Свойство содержит ядро состояния и имеет методы для интерфейсов вызова.

Компоненты могут подписаться на информацию, которая недоступна в основном состоянии. Подписки выполняются через API-интерфейс websocket, который поддерживает синхронизацию данных с серверной частью.

Используется однонаправленный поток данных. Когда вносятся изменения в бэкэнд (например, включается свет), hass объект будет обновлен в корне приложения и станет доступным для каждого компонента, который в нем нуждается.

Поток данных. Интерфейс использует децентрализованную маршрутизацию. Каждый компонент знает о маршрутизации достаточно, чтобы знать, как обращаться с той частью, за которую он отвечает. Дальнейшая маршрутизация передается вниз по дереву компонентов.

Маршрутизация. Например, главный компонент будет смотреть на первую часть URL-адреса, чтобы решить, какая панель должна быть загружена. Каждая панель может иметь собственное сопоставление между URL-адресом и отображаемым контентом.

Заключение. Правильным подходом при разработке приложения является использование современных технологий, которые позволяют решать свои задачи. Это, в первую очередь, экономит время при разработке интерфейса, а также ресурсы на обработку данных. Это является очень актуальным, при большом количестве информации, которая подлежит постоянному изменению.

Рассмотренная технология позволяет создать адаптивный интерфейс, любого уровня сложности, при этом сохраняя его функциональные возможности и привлекательный внешний вид. По итогам данной работы, была описана возможность их применения к построению интерфейса.

ЛИТЕРАТУРА

1. Официальный сайт справочной информации. [Электронный ресурс]. Режим доступа http://asupro.com/building/technology/structure-building-automation-system-bms.html. Дата доступа – 20.09.2020 г.
2. Официальный сайт справочной информации. [Электронный ресурс]. Режим доступа https://ipap.ru/poleznye-stati/4-useful/yazyk-programmirovaniya-python. Дата доступа – 19.09.2020 г.