Также вы имеете возможность выполнить программную проверку политик в режиме Razor. Для этого вы должны сначала ввести зависимость от службы авторизации. Использование службы авторизации в представлении может помочь скрыть элементы пользовательского интерфейса, которые не должны находиться в пределах досягаемости текущего пользователя, учитывая текущий контекст. Однако имейте в виду, что просто скрывать варианты в представлении недостаточно. Вы всегда должны применять политики в контроллере.

Базовые требования политики в основном охватывают претензии(claims), аутентификацию и обеспечивают универсальный механизм настройки на основе утверждений, но вы также можете создавать пользовательские требования. Требование к политике состоит из двух элементов: класса требований, содержащего только данные, и обработчика полномочий, который проверяет данные пользователя. Пользовательские требования расширяют ваши возможности настройки определенной политики.

Требование должно иметь по крайней мере один обработчик авторизации. Обработчик имеет тип AuthorizationHandler<T>, где T - тип требования.

Обработчик требований авторизации считывает претензии, связанные с конкретным пользователем, и проверяет на соответствие с собственными претензиями. Если всех соответствий не найдено, обработчик возвращает отрицательный результат. Положительный результат возвращается только в том случае, если претензии существует и содержит указанное значение. Предполагается, что пользовательское претензии будут частью информации, связанной каким-то образом с пользователем, например, столбцом в одной из таблиц базы данных.

В свою очередь обработчик авторизации вызывает метод Succeed, передавая текущее требование, чтобы уведомить, что требование было успешно проверено. Если это требование не прошло валидацию, обработчик не должен ничего делать и может просто вернуться к дальнейшей работе. Однако, если обработчик хочет определить отказ требования независимо от того, что другие обработчики поэтому же требованию могут завершиться успешно, он вызывает метод Fail на объекте контекста авторизации.

Также, вам необходимо зарегистрировать новый обработчик с системой DI в рамках типа обработчика IAuthorization.

Каждое требование может иметь несколько обработчиков. Когда несколько обработчиков зарегистрированы в системе DI для одного и того же требования достаточно, чтобы по крайней мере один был завершен успешно.

Авторизация на основе политики — это новый подход, который обеспечивает более богатую и выразительную модель. Это связано с тем, что политика представляет собой набор требований на основе требований и пользовательской логики на основе любой другой информации, которая может быть введена из контекста HTTP или внешних источников. Эти требования связаны с одним или несколькими обработчиками, которые отвечают за фактическую оценку требования.

Список используемых источников

1.Документация ASP.NET Core. [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/aspnet/core. – Дата доступа: 20.03.2018.

2.Документация .NET [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/library – Дата доступа: 23.03.2018.

3.Документация и исходный код. [Электронный ресурс]. – Код доступа: https://github.com/aspnet . – Дата доступа: 15.03.2018.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ УПРАВЛЕНИЯ АДМИНИСТРАТИВНОЙ ПАНЕЛЬЮ MAGENTO

Институт информационных технологий БГУИР,
г. Минск, Республика Беларусь

Черный М.С.

Пачинин В. И. – зав. кафедрой ИСиТ, к.т.н., доцент
Сечко Г. В. – доцент каф. ИСиТ, к.т.н,, доцент

Объектом исследования является программное обеспечение выполняющее функцию управления административной панелью Magento с мобильного устройства. В административной панели Magento располагается информация о пользователях ресурса, продуктах, заказах и транзакциях. Также ее главной сильной стороной является обилие встроенных функций: речь идет о валюте, языках, скидках и купонах и многом другом.

Для любой системы онлайновой торговли очень важно обеспечить непрерывность функционирования (отсутствие простоев и замедлений в работе). По сути, это является ключевым требованием, которое крайне важно, как с точки зрения маркетинга и PR, так и с точки зрения бизнеса в целом.

Целью проекта является разработка программного обеспечения для управления административной панелью Magento, с целью своевременного обнаружения неполадок в работе сервера и получения актуальной информации о текущем состоянии аппаратного обеспечения, товарах, пользователях и отчетах за выбранный период.

Для выполнения поставленной задачи было разработано клиент-серверное программное средство, состоящее из iOS приложения, а также серверной части, представленной в виде Magento и Gateway серверов. Архитектура программного средства представлена на рисунке 1.

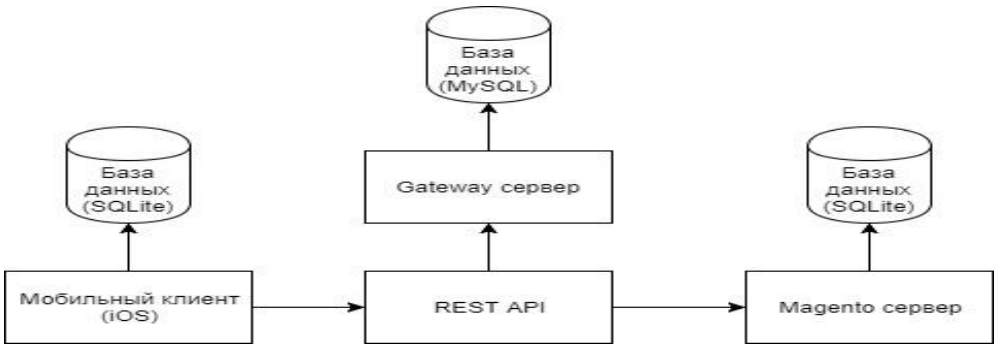

Рисунок 1 – Архитектура программного средства

Каждый из описанных элементов обладает собственной базой данных. Коммуникация между клиентской и серверной частью осуществляется по протоколу HTTPS с использованием сообщений в формате JSON.

Веб-сервер программного средства разработан с применением архитектурного стиля REST, который представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой системы. Разработка серверной части велась на языке программирования PHP.

В качестве СУБД для мобильного приложения была выбрана SQLite, доступ к базе данных осуществляется через CoreData.framework разработанного компанией Apple Inc.

Core Data – это мощный и гибкий фреймворк для хранения и управления графом модели. Основные критерии, обуславливающий данный выбор:

– высокая производительность;
– возможность расширяемости;
– быстродействие и энергоэффективность.

Для разработки мобильного приложения использовалась IDE XCode с использованием языка Swift. Swift – это надежный и интуитивно понятный язык программирования, при помощи которого можно создавать приложения для iOS, Mac, Apple TV и Apple Watch.

Разработанное программное средство может работать как на планшетах, так и на смартфонах под управлением операционной системы iOS.

Мобильное приложение реализует следующие функции:

– предоставление информации о пользователях, продуктах, заказах;
– рассылку SMS сообщений и push-notification уведомлений;
– предоставление данных о аппаратном обеспечении и исключительных ситуациях;
– генерацию отчетов по заданным критериям.

Список использованных источников.

1. Коннолли, Т. Базы данных: проектирование, реализация и сопровождение [текст] / Т. Коннолли. – М. 2001.
2. Нахавандипур, В. IOS. Разработка приложений для iPhone, iPad и iPod / В. Нахавандипур; Пер. с англ. О. Сивченко. – СПб.: Питер, 2013.

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ВЗАИМООТНОШЕНИЯМИ С КЛИЕНТАМИ

Институт информационных технологий БГУИР,
г. Минск, Республика Беларусь

Чернявский С.В.

Бакунова О.М. - ст. преподаватель каф. ИСиТ, м.т.н.

Система управления взаимоотношениями с клиентами или CRM-система - это программное обеспечение для организаций, предназначенное для автоматизации процессов взаимодействия с клиентами. Целью внедрения данной системы является грамотное построение и оптимизация бизнес-процессов организации, тем самым повышение уровня продаж и улучшение обслуживания клиентов. Достигается это путем создания хранилища информации о клиентах, сохранения истории взаимоотношений с ними, планирования и анализа маркетинговых мероприятий, оптимизации работы сотрудников путем фиксации конкретных задач по клиенту.

Характерными признаками необходимости внедрения CRM-системы являются: наличие постоянно растущего списка клиентов, наличие расширенного спектра товаров и услуг, сложный процесс доведения продукта или услуги до конечного клиента, постоянно меняющиеся условия рынка и высокая конкуренция, потребность в прозрачности управления организацией.

На рынке существует достаточное количество готовых решений, которые можно внедрить в организацию. Вместе с этим присутствуют организации со специфичными бизнес-процессами и определенными