Здравствуйте

**Что**

В ходе дипломного проекта был разработан программный модуль формирования электронной очереди.

Разрабатываемый в ходе дипломного проекта программный модуль предназначен для работы с электронными очередями. В отличие от рассмотренных аналогов систем управления очередями, разрабатываемый модуль предназначен для использования внутри организаций для управления потоками посещений руководителей и заместителей руководителей.

Предполагаемых пользователей модуля можно разделить на следующие типы:

* рядовой сотрудник;
* руководитель;
* заместитель руководителя;
* делопроизводитель или секретарь;
* администратор.

Базовые функции модуля, доступ к которым имеют все пользователи, включают:

* просмотр доступных очередей;
* постановка в очередь;
* просмотр вхождений в очереди и текущих позиций в этих очередях;
* просмотр и изменение собственной учетной записи.

В дополнение к базовым функциям, руководители и заместители руководителей имеют доступ к следующим функциям:

* просмотр собственной очереди;
* удаление вхождений в собственную очередь;
* изменение порядка следования сотрудников в очереди;
* предоставление доступа к очереди делопроизводителям;
* отзыв доступа к очереди делопроизводителям.

При постановке в очередь, позиция в очереди определяется с учетом приоритетов. При этом, заместители руководителей имеют приоритет высший, чем рядовые сотрудники; руководители имеют наивысший приоритет.

Владельцы очередей (руководители и заместители руководителей) имеют возможность предоставлять доступ к своей очереди делопроизводителям. Делопроизводитель, обладающий доступом к очереди, может просматривать и управлять очередью аналогично владельцу очереди.

Администраторы имеют доступ имеют ко всем функциям, доступным сотрудникам, а также:

* изменение очередей;
* просмотр учетных записей;
* изменение свойств учетной записи;
* изменение типов учетных записей.

Доступ в систему осуществляется с помощью логина и пароля. Логин и пароль указывается пользователем при регистрации. Пароль может в последствии изменяться пользователем. Также, администратор может управлять паролем. Клиентское приложение обладает функцией сохранения логина и пароля после успешного входа в систему, и последующей вставки в соответствующие поля.

**Зачем**

Потребность в разработке программного модуля формирования электронной очереди, обусловлена жесткой конкуренцией на рынке информационных технологий и необходимостью обеспечить более высокую производительность труда, большую надежность и достоверность информации, лучшую ее сохранность.

Таким образом, применение разрабатываемого программного модуля формирования электронной очереди является целесообразным и необходимым в современных условиях.

Системы управления очередью помогают избежать скопления людей в местах приёма посетителей и организовать «цивилизованный» порядок обслуживания клиентов.

Согласно проведенному технико-экономическому обоснованию разработанная система является экономически эффективной. По расчетам выявлено, что все дополнительные капитальные затраты на освоение, сопровождение и адаптацию нового ПО окупятся в течение трех лет.

**Как**

Разрабатываемый в ходе данного дипломного проекта программный модуль базируется на технологии Microsoft .NET Framework. .NET Framework – это программная платформа для построения приложений на базе семейства операционных систем Windows.

C# – простой, современный, объектно-ориентированный и типобезопасный язык программирования. C# отвечает требованиям спецификации Common Language Specification, рассмотренной выше, что обеспечивает его совместимость с .NET Framework.

Клиент-серверная архитектура

В основе программного модуля лежит модульная архитектура.

Разрабатываемый программный модуль разбит на подмодули, представленные в виде сборок .NET Framework. Сборки в .NET составляют основную единицу развертывания, управления версиями, повторного использования, областей действия активации и разрешений безопасности.

Структурно можно разделить на подмодули: 4 подмодуля

Модуль Data

Данный модуль содержит классы и интерфейсы компонентов работы с данными (БД) на основе Entity Framework, классы для представления сущностей модели данных (пользователь, должность и т.п.).

Модуль Infrastructure

Модуль Infrastructure содержит классы и интерфейсы, общие для всех частей приложения и использующиеся для передачи данных между частями.

Серверная часть приложения, содержащаяся в модуле Services, представляет собой RESTful сервис на основе ASP.NET Web API.

ASP.NET Web API – технология для создания служб HTTP для широкого диапазона клиентов, включая браузеры и мобильные устройства. Веб-API ASP.NET идеально подходит для разработки RESTful приложений на платформе .NET Framework. Позволяет создавать сетевые API-интерфейсы, которые поддерживают большое количество различных типов содержимого, в том числе XML, JSON.

Для хранения данных была спроектирована модель данных. (Описать модель по плакатам)

Работа с базами данных в описываемом программном модуле осуществляется с помощью технологии Entity Framework.

Entity Framework позволяет работать с данными в форме специфических для домена объектов и свойств, таких как клиенты и их адреса, без необходимости обращаться к базовым таблицам и столбцам базы данных, где хранятся эти данные. Entity Framework дает разработчикам возможность работать с данными на более высоком уровне абстракции, создавать и сопровождать приложения, ориентированные на данные, используя меньше кода, чем в традиционных приложениях.

Для использования конечными пользователями предназначено настольное клиентское приложение, содержащееся в модуле Client. Клиентское приложение, входящее в состав описываемого программного модуля, основано на технологии WPF.

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов [12].

При работе с WPF широко используется шаблон MVVM или Model-View-ViewModel. MVVM позволяет отделить логику приложения от визуальной части (представления). (Показать на плакате)

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление – это XAML-код, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели.

Итогом применения шаблона MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

При тестировании разработанного программного модуля, основное внимание уделялось модульному тестированию.

Модульное тестирование позволяет проверить на корректность отдельные модули исходного кода программы. Модульное тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности (модули программ, объекты, классы, функции и т.д.). Обычно модульное тестирование проводится, вызывая код, который необходимо проверить, и при поддержке сред разработки, таких как фреймворки для модульного тестирования или инструменты для отладки.

Цель модульного тестирования – изолировать отдельные части программы и показать, что эти части работоспособны.

После сборки компонентов, программный модуль был подвергнут ручному функциональному тестированию в соответствии с описанными выше требованиями.

В результате тестирования были выявлены дефекты, которые в последствии были устранены. Функциональное тестирование подтвердило соответствие разработанного программного модуля требованиям.