Литература

1 Национальный статистический комитет Республики Беларусь [Электронный ресурс]. – Режим доступа: http://www.belstat.gov.by/ – Дата доступа: 12.04.2017.

УДК 004.7

Е. Ю. Евлампьев, М. И. Жадан

СОЗДАНИЕ НАСТОЛЬНОГО ПРИЛОЖЕНИЯ ПО УЧЕТУ РАБОЧЕГО ВРЕМЕНИ НА ПЛАТФОРМЕ .NET: WCF, WPF

Статья посвящена разработке системы учета рабочего времени, тесно интегрированной в систему управления проектами. Предлагаемая система имеет многоуровневую архитектуру, что обеспечивает хорошую масштабируемость и конфигурируемость. Были разработаны доменные модели и созданы необходимые сервисы. Для создания приложения использован стек технологий .NET. Общее ядро реализовано с использованием технологии WCF. Доступ к базе данных MS SQL Server осуществлен с использованием Entity Framework. Пользовательское приложение построено на WPF, с применением паттернов разработки MVVM.

Система учета рабочего времени – это комплекс программно-аппаратных средств, для контроля, учета и оценки эффективности работы персонала в рабочее время. Система учета рабочего времени является необходимым инструментом для построения максимально эффективной работы персонала. Руководители предприятий заинтересованы в повышении дисциплины своих сотрудников, тогда дисциплинированный сотрудник на рабочем месте будет максимально вовлечен в трудовой процесс, а значит, использовать весь свой интеллектуальный ресурс для производства товаров или услуг, повышая прибыль и улучшая имидж компании.

Сегодня существует достаточно сложные системы и все они, как правило, требуют больших денежных вложений от потребителей. Для малых предприятий этот пункт зачастую является непреодолимым. Учитывая вышеизложенные недостатки, было решено разработать собственную систему учета времени. Приложение могут использовать фрилансеры, бухгалтера, юристы. Основной областью применения являются IT-проекты.

Для создания приложения использован стек технологий .NET. Общее ядро реализовано по технологии WCF. Доступ к базе данных MS SQL Server будет реализован с использованием Entity Framework. Пользовательское приложение построено на WPF, с применением паттернов разработки MVVM. Для построения сервисов использовался программный фреймворк Windows Communication Foundation [1,2]. Клиентские приложения разработаны на технологии Windows Presentation Foundation.

Технология WPF (Windows Presentation Foundation) является частью экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов [3]. Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML [4]. Однако, стоит учитывать, что по сравнению с приложениями на Windows Forms объем программ на WPF и потребление ими памяти в процессе работы в среднем несколько выше. Но это компенсируется более широкими графическими возможностями и повышенной производительностью при отрисовке графики.

Ниже описан процесс создания уровня приложения, лежащего над уровнем доступа к данным – доменная логика. Здесь должна быть сосредоточена вся логика работы приложения – от авторизации пользователей до предоставления конечных точек

подключения к сервисам. На следующем этапе было выделено несколько конечных сервисов: UserService – для управления пользователями; StorageService – для управления хранилищами данных и ProjectTaskService – для управления проектами, заданиями.

Вся логика приложения заключена именно в сервисах. Для авторизации использовалась своя имплементация RoleProvider, необходимого WCF, кроме того, было добавлено логирование для всех операций с сервисами. Для развертывания WCF приложения был реализован хостинг в Windows Service. К этому этапу уже созданы контракты данных и сервисов. В качестве привязки использован WsHttpBinding. Кроме того, для каждого сервиса была определена конечная точка с привязкой MexHttpBinding, которая определяет получения метаданных сервиса, туда заложен стандартный контракт IMetadataExchange.

Для создания настольного клиентского приложения выбрана технология WPF, которая предоставляет широких спектр возможностей для разработки удобного, современного, дружественного пользователю приложения. Далее описан процесс разработки настольного приложения и продемонстрированы его основные возможности.

Для упрощения процесса разработки использован MVVM фреймворк Caliburn.Micro. Разработка производилась по принципу ViewModel First, где представление создается после вью-модели. Паттерн MVVM позволяет полностью отделить слой графического интерфейса и логику приложения.

Основной вью-моделью является MainViewModel, являющейся наследником от Conductor <IView>. Collection. OneActive. Она содержит в себе набор вью-моделей, из которых одновременно является активной только одна. Основная модель является хэндлером некоторых сообщений, которые возникают в ответ на возникающие в других моделях сообщения.

Для авторизации и аутентификации пользователей использовался собственный IAuthenticationManager. Для всех моделей, редактирование которых возможно пользователем, был добавлен механизм валидации. Вместо использования стандартных средств с атрибутами была использована библиотека FluentValidation.

Неавторизованного пользователя встречает окно приветствия (рисунок 1). Здесь он может видеть галерею с описанием основных возможностей системы. В правом верхнем углу окна доступна кнопка входа в систему.

На рисунке 2 показана панель авторизации пользователей.

После успешной авторизации пользователю открывается по умолчанию последнее представление (рисунок 3), на котором пользователь ранее завершил работу. Как видно, сменилась также и локализация, специфичная для текущего пользователя. Набор функционала в меню так же, как и в веб версии, зависит от роли пользователя в системе.

Рисунок 1 – Окно приветствия

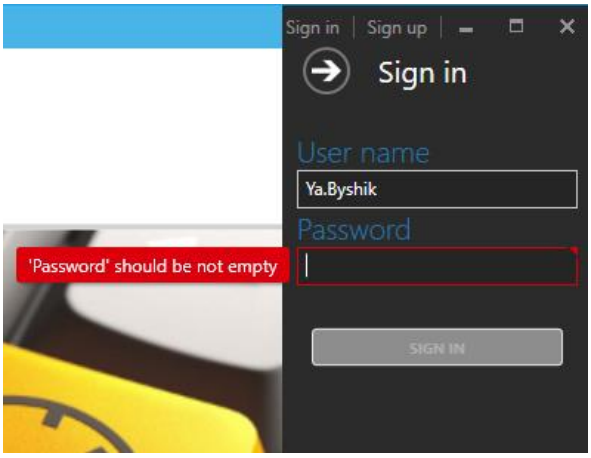

Рисунок 2 – Панель авторизации

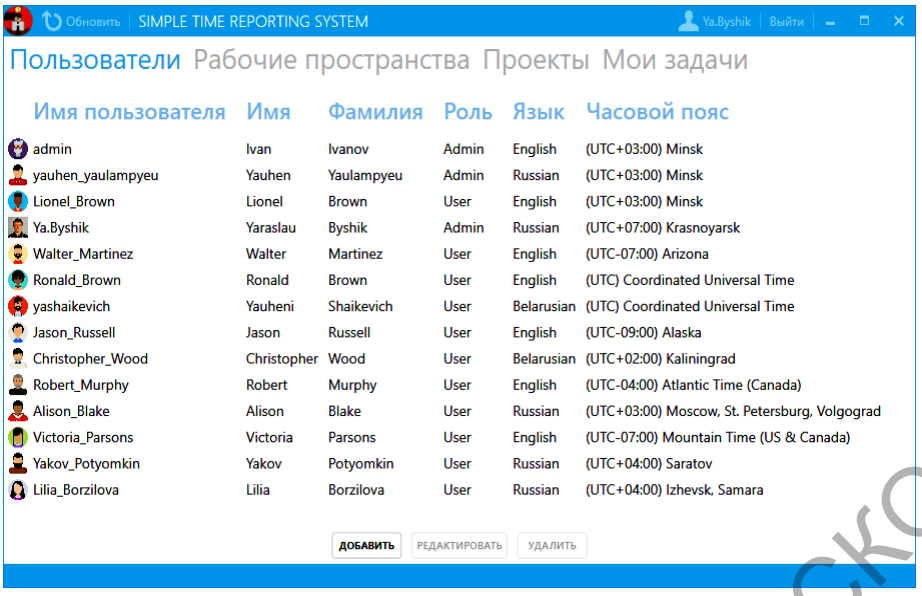

Рисунок 3 – Окно управления пользователями

Пользователю доступны некоторые настройки: фото, отображаемое имя и фамилия, рабочее пространство, предпочитаемый язык интерфейса, часовой пояс. Все эти настройки могут быть изменены из главного окна приложения, вне зависимости от того, с чем работает пользователь.

Для пользователей доступно представление с проектами и задачами к ним (рисунок 4).

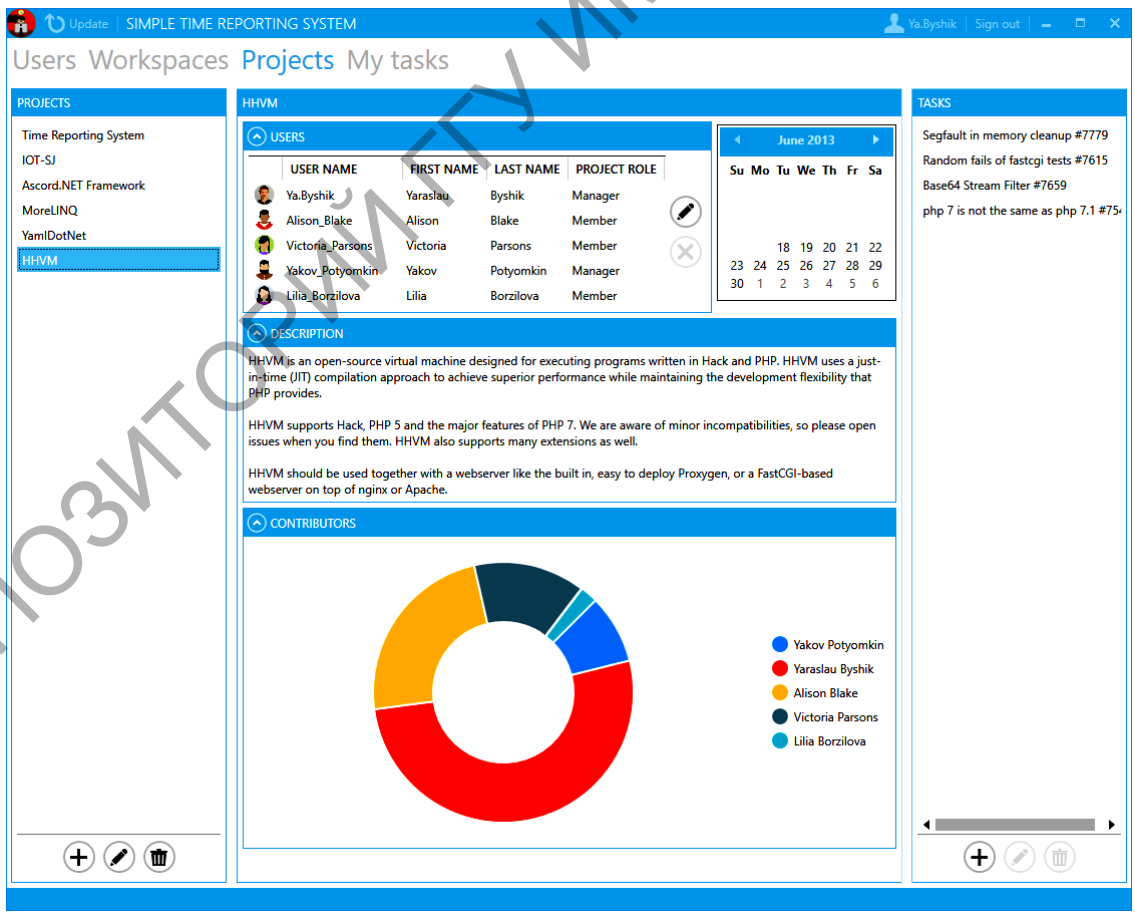

Рисунок 4 – Окно с проектами и задачами

Здесь отображаются только проекты в текущем рабочем пространстве. На левой панели можно выбрать проект, к нему подгружается информация о связанных заданиях, пользователях на нем и описание самого проекта. В окне описания проекта также присутствует диаграмма, показывающая временной вклад в проект каждого пользователя. Редактирование проектов и задач доступно внизу соответствующей панели. Для редактирования используются выпадающие представления.

В информации о задачах отображается календарь и статус задания. Даты соответствуют срокам сдачи задания и проектов.

Менеджеры проекта могут наблюдать за активностью пользователей на задачах. Эта информация отображена на рисунке 5.

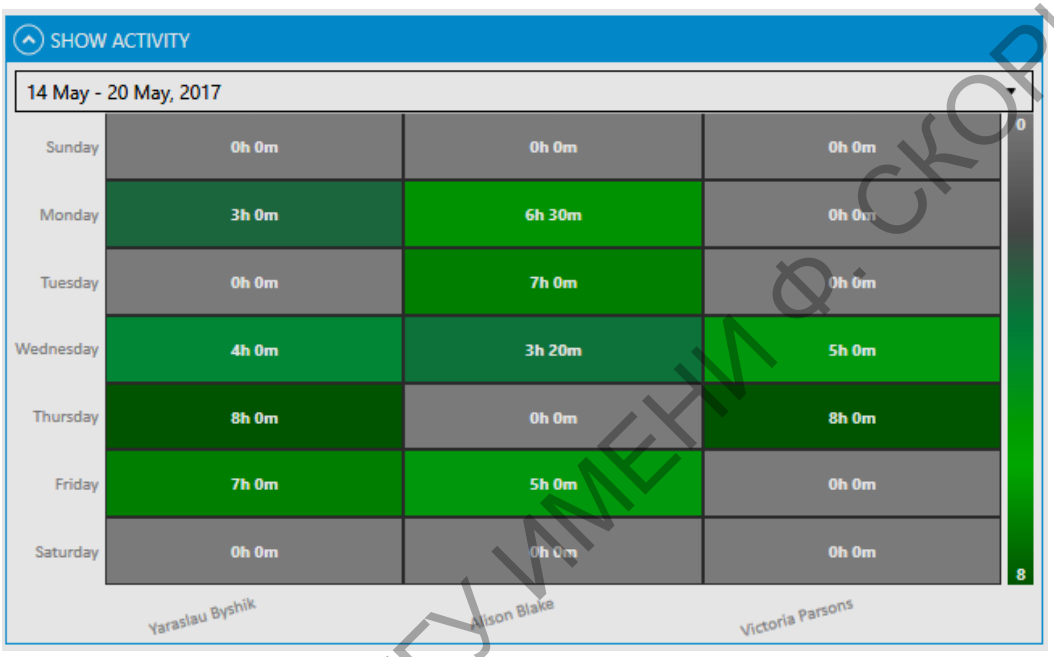

Рисунок 5 – Активность пользователей

Если у пользователя есть активные задания, то он может перейти к ним из меню «Мои задачи». Здесь ему доступно полное описание задачи. Справа находится календарь со сроками. Под описанием доступны текущие отчеты о проделанной работе. К ним с помощью кнопки «+» можно добавить новый, по нажатию на другую кнопку работник может отредактировать выбранный отчет. Также, для удобства, пользователь может отчитаться и задним числом.

Таким образом, реализовано настольное приложение-клиент для взаимодействия с ранее разработанной системой имеет многоуровневую архитектуру. Были наглядно продемонстрированы основные функциональные возможности настольного приложения.

Литература

1 Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – М. : Вильямс, 2013. – 1311 с.

2 Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. / Дж. Рихтер. – СПб. : Питер, 2013. – 4-е изд. – 896 с.

3 Löwy, J. Programming WCF Services, 4th Edition / J. Löwy – S. : O'Reilly Media, 2016. – 1018 с.

4 Lippert, E. Essential C# 6.0, 5th Edition / E. Lippert. – NY : Addison–Wesley, 2015. – 1008 с.