Архитектура проекта изначально подразумевает под собой возможные расширения, такие как мобильные приложения для современных операционных систем и миграцию к одностраничному веб-приложению, поэтому был разработан слой доступа к данным и вынесен в отдельную веб-службу (Web API) на основе фреймворка Grape.

Планирование задач в системе выполняется на основе системы обмена сообщениями RabbitMQ, благодаря которой исчезает проблема жесткой связности модулей, появляется устойчивость к высоким нагрузкам и возможность расширяемости разрабатываемого продукта.

Новизна продукта заключается в том, что до текущего момента не встречались случаи внедрения подобных технологий в белорусскую систему образования, с их последующей интеграцией, базирующейся на принципах простоты, открытости и дружелюбности к использующим его пользователям.

Список использованных источников:

1. RabbitMQ in Action / Manning. - Shelter, New York, 2012.
2. Programming Ruby / The Pragmatic Bookshelf. - Dallas, Texas, 2009.
3. Agile Web Development with Rails / The Pragmatic Bookshelf. - Dallas, Texas, 2014.
4. Grape wiki [Электронный ресурс]. - Электронные данные. - Режим доступа : https://github.com/inridea/grape/wiki.
5. ACM homepage [Электроонный ресурс]. - Электронные данные. - Режим доступа : http://icpc.baylor.edu/welcome.icpc.

ОЦЕНКА ПОДХОДОВ К РАЗРАБОТКЕ КРОССПЛАТФОРМЕННОГО ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Куница Е. Ю., Макоед В. Н.

Искра Н. А. – ассистент

На сегодняшний день существует большое количество инструментов разработки графического интерфейса пользователя (Graphical User Interface, GUI), предлагающих создание программ на различных языках программирования и реализующих различные подходы к созданию интерфейса. Существуют стандартизированные наборы примеров программ, представляющие собой решения типовых задач, возникающих при разработке графического интерфейса [1]. Кроме того, определены метрики, позволяющие выполнить объективное сравнение технологий и подходов к разработке GUI [2].

В рамках исследования были подробно изучены две задачи: CRUD (рис. 1) и Circle Drawer (рис. 2). Были спроектированы и разработаны программы на языках программирования C++ и Java на платформах QT [3] и JavaFX [4] соответственно. Оба подхода являются кроссплатформенными, что позволило успешно протестировать приложения на операционных системах Windows, Linux и OS X. Была произведена оценка скорости и удобства разработки решений на данных платформах.

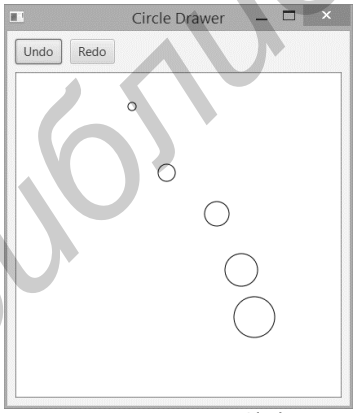

Рис. 1 – окно программы Circle Drawer

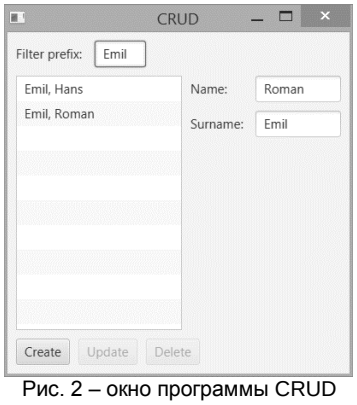

Рис. 2 – окно программы CRUD

При разработке программы для рисования кругов были решены следующие задачи: реализация функциональности отмены действия и повторения отменённого действия (функции Undo и Redo); рисование примитивной графики; работа с диалоговыми окнами.

При разработке CRUD были решены следующие задачи: разделение бизнес-логики и логики интерфейса; реализация изменяющихся элементов интерфейса; нетипичная компоновка элементов интерфейса.

Сходства и различия в возможностях платформ, обнаруженные при разработке приложений, приведены в таблице 1.

Таблица 1. Сравнение возможностей платформ Qt и JavaFX (полужирным шрифтом выделены те особенности, которые оказались более удобными и/или функциональными при разработке)

Критерий сравнения	Qt	JavaFX
Общее для обоих приложений		
Компоновка элементов интерфейса	Класс QLayout (подклассов: 4)	**Класс Pane (подклассов: 13)**
Базовые элементы интерфейса	Базовые элементы интерфейса, характерные для большинства программ разной сложности, присутствуют в обеих библиотеках	
Обработка действий пользователя	Система сигналов и слотов	**Обработка событий, лямбда-выражения**
CRUD		
Представление данных	Класс QTableWidget	**Классы TableView и ObservableList**
Редактирование данных	Работа с объектом класса QTableWidget	Работа с объектом класса ObservableList
Circle Drawer		
Отображение примитивной графики	Класс Canvas	Класс QWidget, метод paintEvent()
Отмена и повторение действий	Реализация не затрагивает библиотеки платформ и использует возможности языков C++ и Java	
Всплывающая кнопка	Базовый класс QWidget	**Специализированный класс Popup**

В рамках данного исследования JavaFX представляется более эффективной и простой в освоении и применении технологией для разработки графического интерфейса пользователя. Библиотека платформы JavaFX более обширна, нежели набор классов, предоставляемый платформой Qt. Поэтому для реализации приложения на Qt может быть характерно увеличение объёма необходимого кода и дополнительные временные затраты на этапе проектирования. Кроме того, механизм сигналов и слотов, применяемый в Qt, оказался менее гибким и интуитивным, нежели обработка событий, доступная при реализации программы на платформе JavaFX. Стоит отметить, что в силу различий языков C++ и Java приложения, реализованные с Qt, менее требовательны к ресурсам, чем аналогичные примеры на JavaFX (таблица 2). Однако различия в процессе разработки, которые и являлись предметом данного исследования, позволяют сделать выбор в пользу JavaFX.

Таблица 2. Сравнение объёмов кода и потребляемых ресурсов (Circle Drawer)

	CRUD		Circle Drawer	
	Qt	JavaFX	Qt	JavaFX
Количество «полезных» строк кода	126	154	323	179
Занимаемая оперативная память при запуске (ОС: Windows 8.1 x64)	4,6 Мб	69 Мб	5 Мб	50 Мб

Список использованных источников:

1. https://github.com/eugenkiss/7guis/wiki // A Notational Usability Benchmark for GUI Programming
2. http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/ // Cognitive Dimensions of Notations Resource Site
3. http://www.qt.io/ // Qt. Cross-platform application & UI development framework
4. http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm // JavaFX Overview (Release 8)