УДК 004.4

ПРИНЦИПЫ И ТЕХНОЛОГИИ РАЗРАБОТКИ ФАЙЛОВОГО МЕНЕДЖЕРА ДЛЯ ВЕБ

Кокурин А.В.[a], Гедранович В.В.[b]

[a] *Минский инновационный университет, магистрант, bear-sasha1@yandex.ru*
[b] *Минский инновационный университет, кандидат педагогических наук, доцент, проректор по научной работе, gedrvv@gmail.com*

Аннотация
Рассматриваются принципы и технологии проектирования файловых менеджеров для веб. Представлен обзор существующих решений файловых менеджеров, предложена технология разработки модульного файлового менеджера для веб.

Ключевые слова: файловый менеджер, файлы, работа с файлами, веб файловый менеджер.

Веб: http://library.miu.by/journals!/item.science-xxi/issue.5/article.12.html

Поступила в редакцию: 31.10.2016.

PRINCIPLES AND TECHNOLOGY OF FILE MANAGER DEVELOPMENT FOR WEB

Kokuryn A.[a], Hedranovich V.[b]

[a] *Minsk Innovation University, Master's degree student, bear-sasha1@yandex.ru*
[b] *Minsk Innovation University, PhD in Pedagogic sciences, Assosiate Professor, vice-rector for research, gedrvv@gmail.com*

Abstract
The principles and technologies of file manager development for web are considered. A review of existing solutions of file managers is presented, the technology of the modular design for the web file manager is suggested.

Keywords: file manager, files, work with files, web file manager.

Web: http://library.miu.by/journals!/item.science-xxi/issue.5/article.12.html

Received: 31.10.2016.

Введение

Для решения задач хранения файлов, организации доступа к ним и управления ими существуют файловые менеджеры.

Файловые менеджеры – это класс программ, служащих для всевозможной работы с файлами, включая поддержку таких операций, как создание, редактирование, копирование и удаление, для обеспечения гибкого и удобного запуска других программ, предназначенных для работы с этими файлами. Нередко они сопровождаются дополнительными утилитами, облегчающими жизнь пользователя. Для многих пользователей любимый файловый менеджер зачастую выступает в роли оболочки, заменяя часть стандартных средств работы с файлами, имеющихся в операционной системе [1].

Выделяют различные типы файловых менеджеров: навигационные или пространственные и двухпанельные. К первому относятся Explorer (Проводник) операционной системы Windows, Finder для Mac OS X, Dolphin для Ubuntu и другие. К двухпанельным файловым менеджерам относятся Far Manager, Total Commander, Double Commander, Free Commander и другие.

Файловые менеджеры используются в операционных системах, а также в веб-среде. Что касается файловых менеджеров для веб, то они наиболее актуальны для применения в профессиональной деятельности контент-менеджеров новостных и медиапорталов, где информация обновляется постоянно.

Обзор исследований по тематике

Большинство файловых менеджеров жестко привязаны к операционной системе, на которой установлено веб-приложение. Поэтому одной из главных проблем в выборе файлового менеджера можно назвать зависимость от операционной системы.

На данный момент в интернете можно найти большое количество файловых менеджеров. При работе с отдельными файловыми менеджерами были выявлены следующие недостатки:

– действия с файлами, в частности, копирование и перемещение, производятся в разных окнах браузера или его вкладках;

– медленная работа, особенно с большим количеством файлов. Не используют кеш браузера, HTML5 функциональности. JavaScript используют только для обновления DOM-модели и AJAX-запросы;

– несинхронная работа пользователей. Если с одним и тем же файлом или папкой работают несколько пользователей, то процент появления критической ошибки возрастает с геометрической прогрессией.

Файловые менеджеры необходимо сделать более «дружественными» для всех категорий пользователей. Интерфейс должен быть хорошо продуман, он должен учитывать большинство UX-правил, которые проверены годами на практике.

Сравнение аналогов

Одним из файловых менеджеров для веб (file manager for web) является elFinder v2.0 (рисунок 1) [2].

Данный файловый менеджер первоначально был реализован на языке программирования php – скриптовый язык программирования общего назначения, интенсивно применяемый для разработки веб-приложений. А затем программистами из разных стран мира были написаны так называемые коннекторы, которые использовали одну реализацию на клиенте, а серверная часть приложения была написана на языке, с которым непосредственно работал сам сервер, – это было реализовано путем «открытости», то есть проект доступен в публичном доступе на GitHab [3]. Решение с «фиксированной» клиентской стороной является оптимальным, что позволяет достичь единого описания клиентской библиотеки и в дальнейшем быстрого ее обновления – это является важнейшим плюсом данного файлового менеджера.

Другим немаловажным достоинством данного файлового менеджера является большой объем методов: создание папок, создание, перемещение, удаление файлов, предпросмотр мультимедиа-файлов и так далее.

К минусам elFinder'а можно отнести прекращение его поддержки и усовершенствования, так как последняя версия не является стабильной и выпущена 10 апреля 2012 года, что означает использование старых технологий.

Файловый менеджер DevExpress (рисунок 2) [4] – является одним из компонентов всего пакета DevExpress и доступен бесплатно только в течение тестового периода, а в дальнейшем необходимо приобрести лицензионную версию данного программного обеспечения (ПО). При работе с данным файловым менеджером возникает много вопросов, связанных с тем или иным желаемым действием, что вызывает некоторые трудности.

Главным требованием любого файлового менеджера является его скорость, но данное ПО от DevExpress является очень медленным даже при работе с небольшим количеством файлов. Единственной положительной его стороной можно назвать только внешний вид, хотя и у него есть минусы.

Стоит уделить особое внимание файловому менеджеру FileUltimate от компании GleamTech. Внешний вид файлового менеджера FileUltimate (рисунок 3) похож на проводник семейства Windows. Преимущество у этого файлового менеджера заметно сразу – приятный дизайн, быстрая работа и большое разнообразие операций с файлами.

Этот файловый менеджер выделяется из всех вышеперечисленных, но и у него есть минус – вся его работа осуществляется через контролы (элементы управления) технологии WebForms, которая в какой-то степени является устаревшей.

Анализ файловых менеджеров выполнен на основе работы с их демо-версиями, которые открыты в публичном доступе на официальных сайтах.

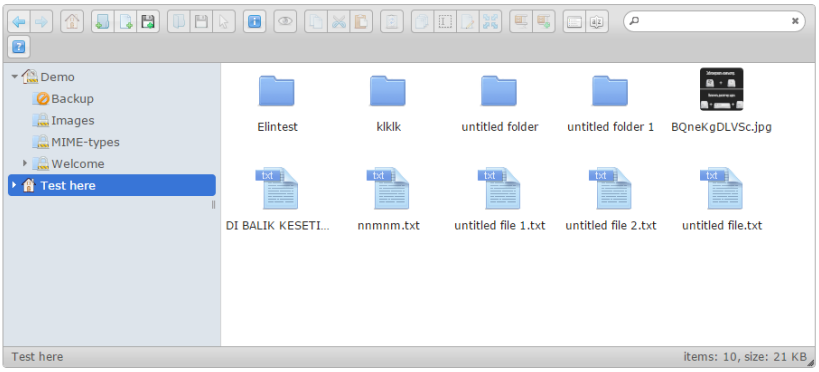

Рисунок 1 – Внешний вид файлового менеджера elFinder v2.0

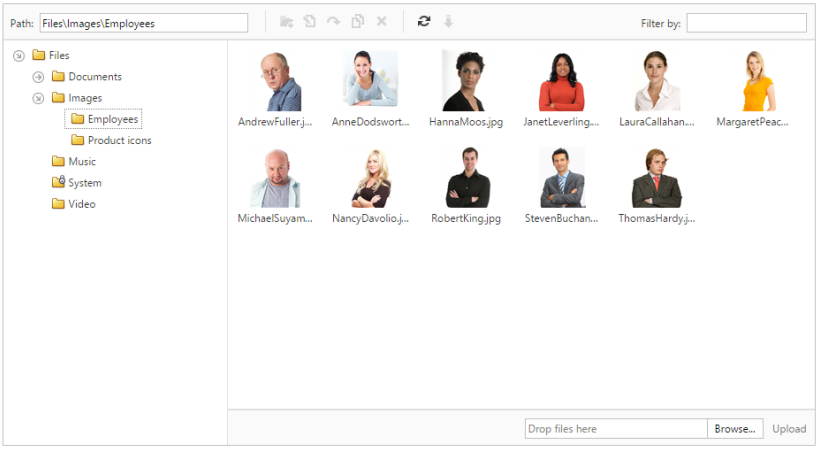

Рисунок 2 – Внешний вид файлового менеджера DevExpress

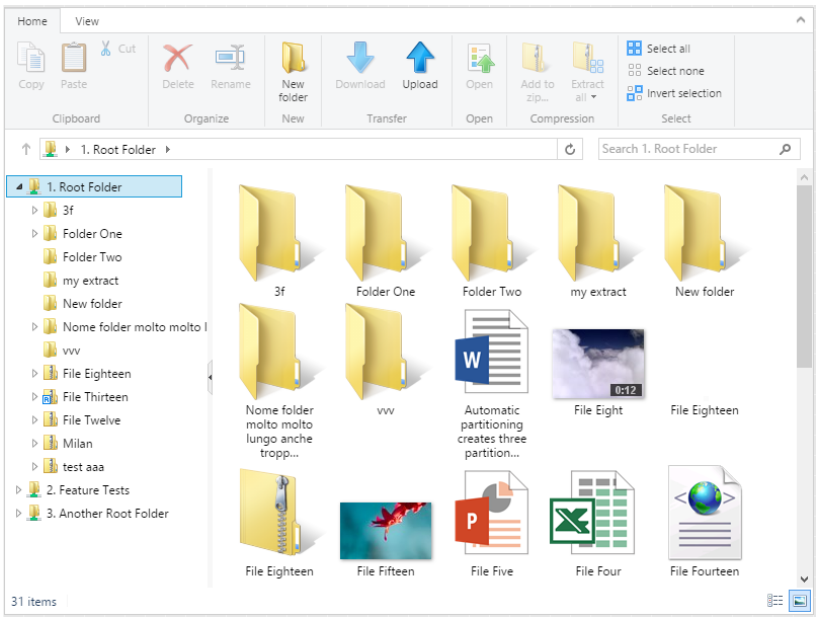

Рисунок 3 – Внешний вид файлового менеджера FileUltimate

Главные задачи любого файлового менеджера – это работа с файлами: создание, копирование, перемещение, удаление, отображение файлового дерева. Все файловые менеджеры, рассмотренные выше, реализуют эти задачи, но скорость работы каждого из них разная. Скорость работы ПО в интернете зависит не только от архитектуры приложения и выбранного языка программирования, но и от параметров, которые не зависят от разработчиков: загруженность сети в определенный момент времени, пропускная способность канала, скорость доступа, предоставляемая провайдером, и другие.

Некоторые из файловых менеджеров предоставляют дополнительные функции, которые позволяют

пользователям какого-либо сайта работать с файлами очень продуктивно и не использовать ftp-клиенты для закачивания файлов на компьютер, их редактирования и загрузки обратно на сервер, что позволяет экономить много времени.

Проектирование файлового менеджера

Одной из важнейших проблем при разработке собственных встроенных плагинов[1], или отдельной части приложения, как файловый менеджер – это использование событий и правильного подхода при построении верстки, которые не должны конфликтовать с разрабатываемым веб-приложением, в который они будут встраиваться.

Задачей на сегодня можно назвать создание файлового менеджера, который будет разрешать пользователю легко работать с файлами, используя всем привычный интерфейс пользователя – кнопки, контекстное меню, технологию drag-&-drop, легкую работу с манипуляцией типов файлов и другое.

Проектируемый файловый менеджер представляет собой библиотеку, которая, в свою очередь, включает в себя достаточно большое количество файлов как исполняемых на сервере, так и исполняемых на клиенте. Данная библиотека рассматривается прежде всего как ресурс с многоуровневой архитектурой, состоящей из многих программных модулей. Важнейшее требование к такой библиотеке – высокая продуктивность в условии больших нагрузок.

Работа с файловым менеджером, с которым работает пользователь, разделена на 4 этапа: прием данных в формате JSON, отображение «псевдо-файлов» на клиенте, обработка событий, которые вызвал пользователь, и отдача данных на сервер.

Файловый менеджер может содержать большое количество файлов и тем самым сильно тормозить систему. Необходимо использовать асинхронные запросы по мере их надобности, «умный» интерфейс, который избавит пользователя от индикаторов загрузки. А реализация паттернов (шаблонов) MVVM [5] и Observer [6], которые будут строить HTML-DOM без дополнительной обработки кода, избавит разработчика от многих проблем.

Программный шаблон проектирования MVVM

Паттерн MVVM (Model-View-ViewModel) – применяется для проектирования архитектуры приложения, который ориентирован на современные платформы разработки.

Данный шаблон проектирования используется для разделения модели и ее представления, что необходимо для изменения их отдельно друг от друга. Например, разработчик задает логику работы с данными, а дизайнер, соответственно, работает с пользовательским интерфейсом.

Удобство использования заметно в тех случаях, когда в платформе, на которой ведется разработка, присутствует «связывание данных» [5].

Сам шаблон состоит из трех частей: Model (Модель), View (Представление) и ViewModel (Модель представления) (рисунок 4).

Модель, как и в классическом паттерне MVC, представляет собой фундаментальные данные, необходимые для работы приложения.

Представлением является графический интерфейс, и одновременно он же является подписчиком на событие изменения значений свойств или команд, предоставляемых **Моделью представления**. Если в Модели представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, а затем Представление запрашивает обновленное значение свойства из Модели представления. А если пользователь воздействует на какой-то определенный элемент графического интерфейса, то Представление вызовет соответствующую команду, предоставленную Моделью представления.

Модель представления является абстракцией Представления с одной стороны, а с другой, предоставляет обертку данных из Модели, которые подлежат связыванию, то есть она содержит Модель, которая преобразована к Представлению, а также содержит в себе команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Программный шаблон Observer (Наблюдатель)

Шаблон проектирования Observer находит широкое применение в системах пользовательского интерфейса, в которых данные и их представления (виды) отделены друг от друга [6]. При изменении данных должны быть изменены все представления этих данных, например, в виде таблицы, графика и диаграммы.

Назначение Observer:

– определяет зависимость «один-ко-многим» между объектами так, что при изменении состояния одного объекта все зависящие от него объекты уведомляются и обновляются автоматически;

– инкапсулирует главный, независимый компонент в абстракцию Subject, а изменяемые, зависимые компоненты – в иерархию Observer.

На рисунке 5 представлена UML-диаграмма, которая иллюстрирует работу программного шаблона Observer.

Этот паттерн используется во многих современных клиентских фреймворках, таких как KnockoutJS, AngularJS и другие.

Кратко описать работу данного паттерна, в основе которого лежат объекты Subject (Субъект) и

[1] **Плаги́н** (англ. plug-in, от plug in «подключать») – независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования ее возможностей. **Плагины** обычно выполняются в виде библиотек общего пользования.

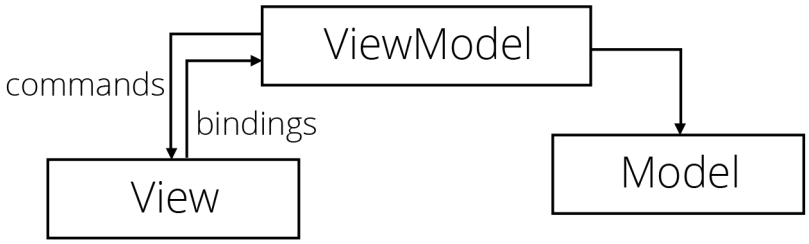

Рисунок 4 – Схема паттерна MVVM

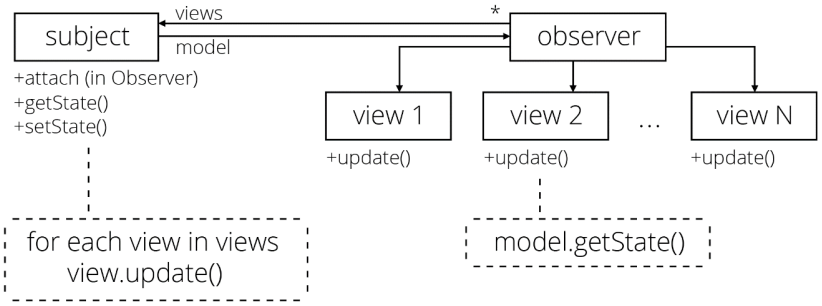

Рисунок 5 – UML-диаграмма классов паттерна Observer

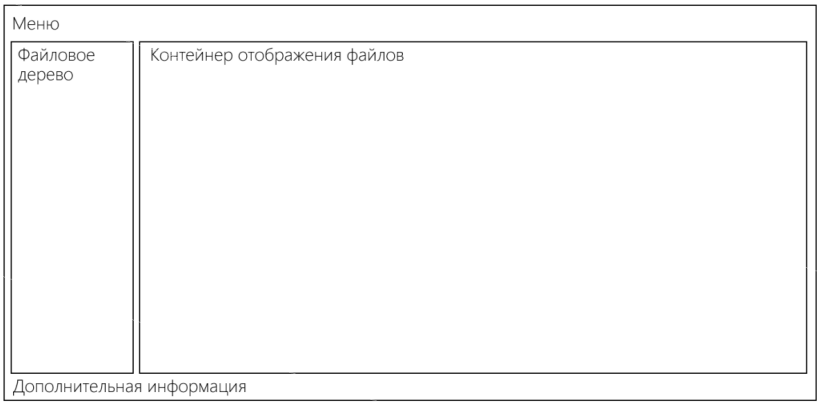

Рисунок 6 – Визуальная структура файлового менеджера

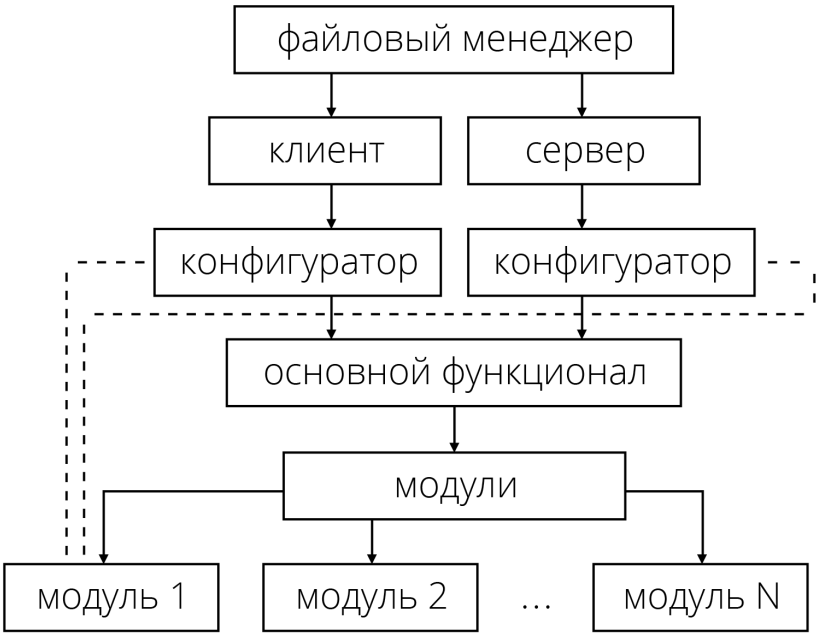

Рисунок 7 – Логическая структура файлового менеджера

Observer (Наблюдатель), можно следующим образом:

– Субъект изменяется и уведомляет о своих изменениях зависимых от него Наблюдателей;

– Наблюдатели синхронизируют свои данные с Субъектом;

– Субъект рассылает уведомления своим Наблюдателям, даже не зная о том, какие объекты ими являются. При этом количество подписчиков не ограничено.

Многие файловые менеджеры, а также плагины для работы веб-приложения используют множество дополнительных библиотек. Это не всегда оправданно. Например, написанная на JavaScript библиотека jQuery [7] имеет достаточно большой размер, однако из всего огромного предоставляемого ей функционала используется лишь около 10 %. По нашему мнению, использование этой библиотеки нецелесообразно, так как можно обойтись javascript-фреймворком VanillaJS, предоставляющим тот же функционал и не требующим никакой загрузки стороннего кода.

Стили являются отдельным вопросом в создании файлового менеджера и других внешних плагинов. Благодаря методологии БЭМ (Блок-Элемент-Модификатор) [8], разработанной в российской компании Яндекс для быстрого создания веб-сайтов, поддерживать которые необходимо долгие годы. Отличительной чертой данной методологии является то, что она позволяет создавать расширяемые и повторно используемые компоненты интерфейса.

При проектировании гибкого программного обеспечения, в данном случае файлового менеджера, необходимо следовать **следующим принципам**: модульность и хорошо проработанный программный интерфейс.

Визуальная структура

Исторически сложилось, что все файловые менеджеры очень схожи. Это является плюсом, так как каждый пользователь персонального компьютера работал с файловым менеджером и делал как минимум операцию по копированию файлов. Поэтому файловый менеджер имеет стандартную структуру (рисунок 6).

Панель «Меню» представляет собой набор элементов управления для работы с файлами, а также быстрые операции с файлами, такие как удаление, создание папки и другие.

Панель «Файловое дерево» отображает папки в древовидной структуре.

Панель «Контейнер отображения файлов» похожа на панель «Файловое дерево», но отображает не только папки, но и файлы в нескольких видах отображения.

Панель «Дополнительная информация» содержит в себе дополнительную информацию об активной папке.

Логическая структура

Данный файловый менеджер предназначен для всех разработчиков и их клиентов, которым необходима работа с файлами в своих проектах.

Этот файловый менеджер имеет необходимый основной функционал, обладает простым и интуитивно понятным интерфейсом, что позволит быстро его освоить даже начинающим пользователям, так как он визуально похож на десктопные файловые менеджеры, к которым все давно привыкли.

Благодаря модульной системе файлового менеджера можно использовать только те модули, которые необходимы в каком-либо определенном проекте, а если понадобится расширить функционал, то это можно будет сделать достаточно легко – «скачать» и установить эти модули в его исходный проект или установить всего одной строчкой через консоль NuGet Package Manager [9] или NPM [10].

Файловый менеджер в веб-среде разделяется на две слабосвязанные структуры – на клиент и сервер, в отличие от системного файлового менеджера. Структура разрабатываемого файлового менеджера представлена на рисунке 7.

Каждая структура, клиентская сторона и серверная, имеют файлы конфигурирования, которые помогают настроить файловый менеджер. Файл конфигурирования web.config на серверной стороне имеет свою секцию в файле настроек приложения, куда непосредственно будет встраиваться этот менеджер, а настройка клиентской стороны осуществляется в файле config.js.

Первоначально файловый менеджер имеет базовый функционал: добавление файлов, копирование, перемещение, удаление и переименование. В дальнейшем если разработчику необходимо использовать дополнительный функционал, то уже установленный файловый менеджер можно расширить разработанными модулями.

Заключение

Анализ готовых решений файловых менеджеров показал, что они не в полной мере удовлетворяют выбору каждого программиста, а пользователи, которые будут использовать выбранные программистами файловые менеджеры, будут долгое время обучаться работе с ними.

Работа файлового менеджера для веб заключается в предоставлении пользователям возможности просмотра файлов на сервере и работы с ними. Правильное и сбалансированное применение принципов, методов и средств разработки файлового менеджера для веб-среды гарантирует успешное решение поставленных задач.

ЛИТЕРАТУРА / REFERENCES

1. Кальченко, Д. Файловый менеджер – инструмент профессионального пользователя [Электронный ресурс] / Д. Кальченко // КомпьютерПресс. – 2005. – № 9. – Режим доступа: http://compress.ru/article.aspx?id=14449. – Дата доступа: 29.07-2016.

Kal'chenko, D. Faylovyy menedzher – instrument professional'nogo pol'zovatelya [Elektronic resource] / D. Kal'chenko // Komp'yuterPress. – 2005. – No. 9. – Mode of access: http://compress.ru/article.aspx?id=14449. – Date of access: 29.07-2016.

2. Studio-42/elFinder: Open-source file manager for web, written in JavaScript using jQuery and jQuery UI web [Electronic resource]. – Mode of access : http://elfinder.org. – Date of access : 15th of August, 2016.

3. elFinder 2.1.x – file manager for web [Electronic resource]. – Mode of access: https://github.com/Studio-42/elFinder. – Date of access : 15th of August, 2016.

4. .NET UI Controls for Developers of Mobile, Desktop, Web – Reporting Applications | www.DevExpress.com [Electronic resource] // ASP.NET AJAX. – Mode of access: https://demos.devexpress.com/ASPxFileManagerAndUploadDemos/FileManager/Features.aspx. – Date of access : 29th of July, 2016.

5. Model-View-ViewModel [Электронный ресурс] // Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Model-View-ViewModel. – Дата доступа: 03.09.2016.

Model-View-ViewModel [Electronic resource]. – Vikipediya. Svobodnaya entsiklopediya. – Mode of access: https://ru.wikipedia.org/wiki/Model-View-ViewModel. – Date of access: 03.09.2016.

6. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма [и др.]. – СПб: Питер, 2001. – 368 с.

Priyemy ob"yektno-oriyentirovannogo proyektirovaniya. Patterny proyektirovaniya / E. Gamma [i dr.]. – SPb: Piter, 2001. – 368 p.

7. jQuery [Electronic resource]. – Mode of access: https://jquery.com. – Date of access : 18th of August, 2016.

8. Методология [Электронный ресурс] // БЭМ. – Режим доступа: https://ru.bem.info/methodology. – Дата доступа: 26.09.2016.

Metodologiya [Electronic resource] // BEM. – Mode of access: https://ru.bem.info/methodology. – Date of access: 26.09.2016.

9. NuGet Gallery | Packages [Electronic resource]. – Mode of access: https://www.nuget.org/packages. – Date of access : 29th of August, 2016.

10. npm [Electronic resource]. – Mode of access: https://www.npmjs.com – Date of access : 29th of August, 2016.