МИНОБРНАУКИ РОССИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информатики
Кафедра теоретических основ информатики (ТОИ)

УДК 004.422.81

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
Зав. каф. ТОИ, к.т.н., доцент
_____ А. Л. Фукс
«_____»_____2014 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ РЕДАКТИРОВАНИЯ ДОКУМЕНТОВ НА ЯЗЫКЕ PILLAR

по основной образовательной программе подготовки бакалавров
010400 – Информационные технологии

Фатеев Алексей Владимирович

Руководитель ВКР,
ассистент кафедры ТОИ
_____ М.А.Филонов
«_____»_____ 2014 г.

Автор работы
студент группы № 1401
_____ А.В.Фатеев

Электронная версия бакалаврской работы
Помещена в электронную библиотеку

Администратор электронной
библиотеки факультета
_____ Е.Н. Якунина

Томск 2014

РЕФЕРАТ

Выпускная квалификационная бакалаврская работа, 34 с., 12 рисунков, 13 источников, 1 приложение.

РАБОТА С ДОКУМЕНТАМИ, ВЕБ-СЕРВИС, SMALLTALK, PHARO, SEASIDE, PILLAR, WEB-ANNOUNCMENT

Объект исследования – способы работы с документами на языке Pillar.

Цель работы – спроектировать и разработать веб-приложение для создания, редактирования, хранения и просмотра документов на языке Pillar.

Метод исследования – практическая работа на ЭВМ.

Основные результаты – спроектирована архитектура приложения, по ней создано веб-приложение, в котором реализованы инструменты для просмотра, создания и редактирования документов на языке Pillar.

ОГЛАВЛЕНИЕ

Введение

В настоящее время для написания статей и документов в среде Интернет распространены так называемые языки разметки – наборы символов, вставляемых в текст для передачи информации о его выводе или строении. Текстовый документ, написанный с использованием языка разметки, содержит не только сам текст, но и дополнительную информацию о различных его участках — например, указание на заголовки, выделения, списки.

Pillar – это язык разметки, который используется для написания документации к среде Pharo Smalltalk.

Для редактирования Pillar документов применяются, как правило, десктопные приложения, которые не поддерживают возможность совместной работы над документами.

Целью данной работы является проектировка и разработка веб-приложения на языке Pharo Smalltalk для создания, редактирования, хранения и сопровождения документов на языке Pillar.

Глава 1. Обзор существующих решений.

Pillar[1] – язык разметки и связанные с ним инструменты, которые используются для оформления различной документации. Этот язык был создан в институте Инриа в рамках проекта Pharo Smalltalk. Основная область применения Pillar - написание книг и статей о среде Pharo Smalltalk и ее составляющих.

Pillar имеет следующие особенности:

- простой синтаксис разметки с поддержкой ссылок, таблиц, рисунков, подписей;

- экспорт в HTML, LaTeX, Markdown, PDF;

- подсветка синтаксиса генерируемых блоков кода

- настраиваемая нумерация заголовков и списков

На рис. 1.1 слева изображен пример синтаксиса документа на языке Pillar. Справа – преобразованный вариант.

Рисунок 1.1 Пример синтаксиса Pillar

Несмотря на некоторую ограниченную область применения языка, существуют различные проекты, позволяющие работать с Pillar-документами.

Одним из распространенных вариантов разработки Pillar-документов является использование сайта https://ci.inria.fr. На этом сайте создается проект, определяется его логическая и физическая структура, задаются правила компоновки единого итогового объекта из составных частей. Каждый участник проекта на десктопном программном обеспечении реализует свою часть и загружает результат в проект. Конечный продукт компилируется после того, как каждая из частей будет реализована и загружена на сервер.

Однако этот подход имеет следующие недостатки: во-первых, ограничена возможность доступа обычных пользователей к результатам работ, что делает эту систему закрытой для большинства. Во-вторых, в основном этот сайт предназначен для создания и ведения объемных проектов, а для редактирования небольших документов нет смысла тратить ресурсы на создание проекта со сложной структурой.

Существует Pier CMS[2] (Content Management System) – система управления веб-контентом, фреймворк для Seaside, устанавливаемый как расширение на образ Pharo. Pier CMS разработана, чтобы предоставить пользователям возможность управлять веб-контентом из любого браузера.

Однако Pier CMS используется, как правило, для управления всем содержимым страницы, и не специализирована на создании отдельных статей.

Так же эта система позволяет только администратору (или ограниченному кругу лиц с правами администратора) создавать и редактировать Pillar-элементы. Эта особенность противоречит концепции открытости. Необходимо, чтобы любой желающий мог создавать и публиковать свой Pillar-контент.

Глава 2. Проектирование приложения

2.1 Описание предметной области.

В данной главе описана концептуальная схема предметной области веб-приложения PillarHub, которая отражает основные сущности и их зависимости. Это описание специально упрощено, чтобы отражать высокоуровневые требования, но в то же время не затрагивать реализацию.

PillarHub – это веб-приложение, предназначенное для создания, редактирования, хранения и просмотра документов на языке Pillar.

PillarHub позволяет работать с двумя типами документов - статьями и книгами. Статья представляет собой сущность, содержащую текст на языке Pillar и метаинформацию, такую как автор, дата публикации и т.д. Книга - это сущность с заранее заданной структурой, состоящая из нескольких упорядоченных статей.

Все документы распределены по хабам. Хаб – это своего рода хранилище для документов. Существует два типа хабов – личные, принадлежащие конкретному пользователю, и общие, принадлежащие группе людей (на момент написания этой работы реализованы только личные хабы). Все хабы находятся в корневой папке, существующей в единственном экземпляре.

Ниже представлена ER диаграмма, иллюстрирующая основные сущности (см. рис. 2.1).

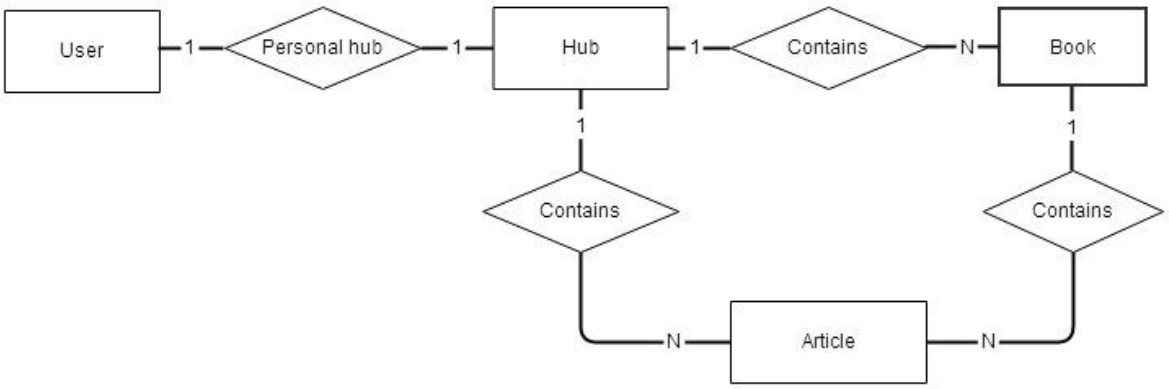

Рисунок 2.1 Упрощенная ER схема модели данных

2.2 Архитектура построения интерфейса приложения

Для построения интерфейса и организации взаимодействия визуальных компонентов была выбрана архитектура фреймворка WebAnnouncement[3].

WebAnnouncment – фреймворк для Seaside-приложений, особенностями которого являются:

1. Использование Model-View-Controller подхода к построению интерфейса (иерархия вью, контроллеры, модель).

2. Слабое связывание компонент при помощи встроенного в среду разработки Pharo фреймворка Announcement.

3. Многоуровневая архитектура контроллеров.

2.2.1 Структура приложения в рамках фреймворка WebAnnouncment.

WebAnnouncement представляет веб-приложение в виде набора объектов трех видов:

a. Компоненты (Component).

b. Лэйауты (Layout).

c. Контроллеры (Controller).

Компоненты – это классы, которые отвечают за отображение данных и различных элементов управления, а так же за отправку аннаусментов, сигнализирующих о действиях пользователя. Компоненты должны проектироваться таким образом, чтобы их можно было тестировать отдельно от всего сайта, то есть все нужные данные между компонентами должны передаваться в виде переменных.

Лэйауты предназначены для расположения компонентов непосредственно на веб-странице. Лэйаут может задавать порядок и способ отображения компонентов, в случае если на одном лэйауте их несколько. Таким образом, контроллер избавляется от необходимости знать об особенностях отрисовки компонентов.

Контроллеры предназначены для инициализации компонентов приложения и размещения их на лэйауте. При необходимости, контроллеры могут прямо во время работы приложения подменять компоненты, отображаемые на лэйауте.

Также контроллеры отвечают за обработку аннаусментов, приходящих от компонентов. Как правило, при получении аннаусмента, контроллер изменяет состояние какого-либо компонента или модель данных всего приложения.

2.2.2 Взаимодействие компонент в рамках фреймворка WebAnnouncement.

При использовании WebAnnouncment все компоненты и контроллеры наследуются, как правило, от класса WAAnnouncingComponent. У каждого такого объекта есть ссылка на экземпляр класса Announcer, реализованная с помощью паттерна lazy-initialization. Анноусер предназначен для передачи и приема аннаусментов между компонентами. Аннаусмент – это объект, наследник класса Announcment. Он может инкапсулировать в себе какое-то сообщение и параметры к нему.

Существует возможность иметь один общий аннаусер на целое дерево компонентов. Это реализовано при помощи метода propogateAnnouncer, который передает аннаусер текущего компонента всем своим дочерним элементам. Это позволяет родительскому контроллеру обрабатывать аннаусменты, приходящие от нижележащих компонентов, не зная при этом, кто и как их сгенерировал.

Такая структура есть следствие применения паттерна Посредник – с помощью одного общего аннаунсера обеспечивается взаимодействие множества компонентов, несвязанных друг с другом напрямую. Таким образом, устраняется сильная связанность и централизуется управление.

Однако может произойти ситуация, в которой какой-нибудь контроллер не будет передавать аннаусер предка своим наследникам, а создаст новый. Таким образом, этот контроллер будет иметь два аннаусера – один для общения с деревом выше, другой для общения со своим поддеревом. Такое построение исключает возможность перемешивания аннаусментов из разных областей видимости, например, когда на одной странице располагаются несколько экземпляров одного класса.

2.2.3 Многоуровневая архитектура контроллеров.

Для удобства управления было решено представить архитектуру в виде нескольких слоев. Каждый слой содержит в себе модули - наборы классов, логически объединенных для решения группы близких по смыслу задач. Понятие «модуль» введено для того чтобы разделить различную логику приложения на обособленные части. Помимо прочих классов, каждый модуль, согласно структуре WebAnnouncement, обязательно должен содержать в себе класс-контроллер и класс-компонент.

На первом уровне располагается Корневой Модуль. Его контроллер отвечает за отображение элементов второго уровня и обработку аннаусментов для перехода между ними.

На втором уровне располагаются модули, семантически представляющие из себя веб-страницу. Каждым из этих модулей будет управлять свой контроллер, отвечающий за логику отображения его компонентов.

Так как модули второго уровня могут состоять из нескольких частей с обособленной логикой, предполагается наличие более глубоких уровней. Можно объявлять все новые и новые уровни архитектуры по мере необходимости.

Глава 3. Программная реализация

Веб-приложение реализуется на Pharo Smalltalk [4][5]. Pharo Smalltalk – объектно-ориентированный язык программирования с динамической типизацией данных.

Для разработки веб-приложений на Pharo используется Seaside [6][7]. Seaside - бесплатный open-source фреймворк, к основным особенностям которого относятся объектно-ориентированный подход к проектированию архитектуры приложения и использование компонентной архитектуры, в которой страницы построены как деревья независимых компонентов, каждый из которых инкапсулирует небольшую часть страницы.

3.1 Реализация модели

В качестве базы данных для хранения экземпляров этих объектов используется SandStoneDB[10][11] – объектная базу данных, предназначенная для использования в небольших приложениях. SandStoneDB хранит объекты на диске в виде коллекций.

Каждый класс, предполагающий сохранение своих объектов в базе данных, должен быть унаследован от SDActiveRecord. SDActiveRecord – это класс, содержащий в себе поля и методы, необходимые для взаимодействия с базой данных. Рассмотрим, как реализуется взаимодействие на примере частовыполняемых операций:

- метод «save» сохраняет объект в базу данных, если тот не противоречит ограничениям целостности. Пример:

```
PHArticle new; save.
```

- метод «remove» удаляет объект из базы данных. Пример:

```
oldArticle remove.
```

- метод «findAll: aBlock» возвращает все объекты, удовлетворяющие условию aBlock. Пример:

```
PHArticle findAll: [:a | a author = ‘Alexey Fateev’].
```

Для унификации предметной области было решено реализовать схему, которая похожа на файловую систему. Это значит, что существуют композитные и листовые объекты. У каждого элемента есть путь, состоящий из списка объектов-родителей. Каждый дочерний объект определяется своим идентификатором – именем, уникальным в рамках родительского объекта. То есть у статьи с именем article1, которая лежит в хабе alexeyfateev, будет путь alexeyfateev/article1.

Нам необходимо одинаково взаимодействовать как с простыми объектами (статьями), так и с составными (хабы, книги), состоящими из нескольких элементов. А так же необходимо будет представлять составные элементы в виде иерархического дерева. Спроектированная структура решает обе проблемы за счет применения паттерна Компоновщик[8][9].

Так как используется объектная база данных, модель предметной области реализуется сразу в классах (см. рис. 3.1):

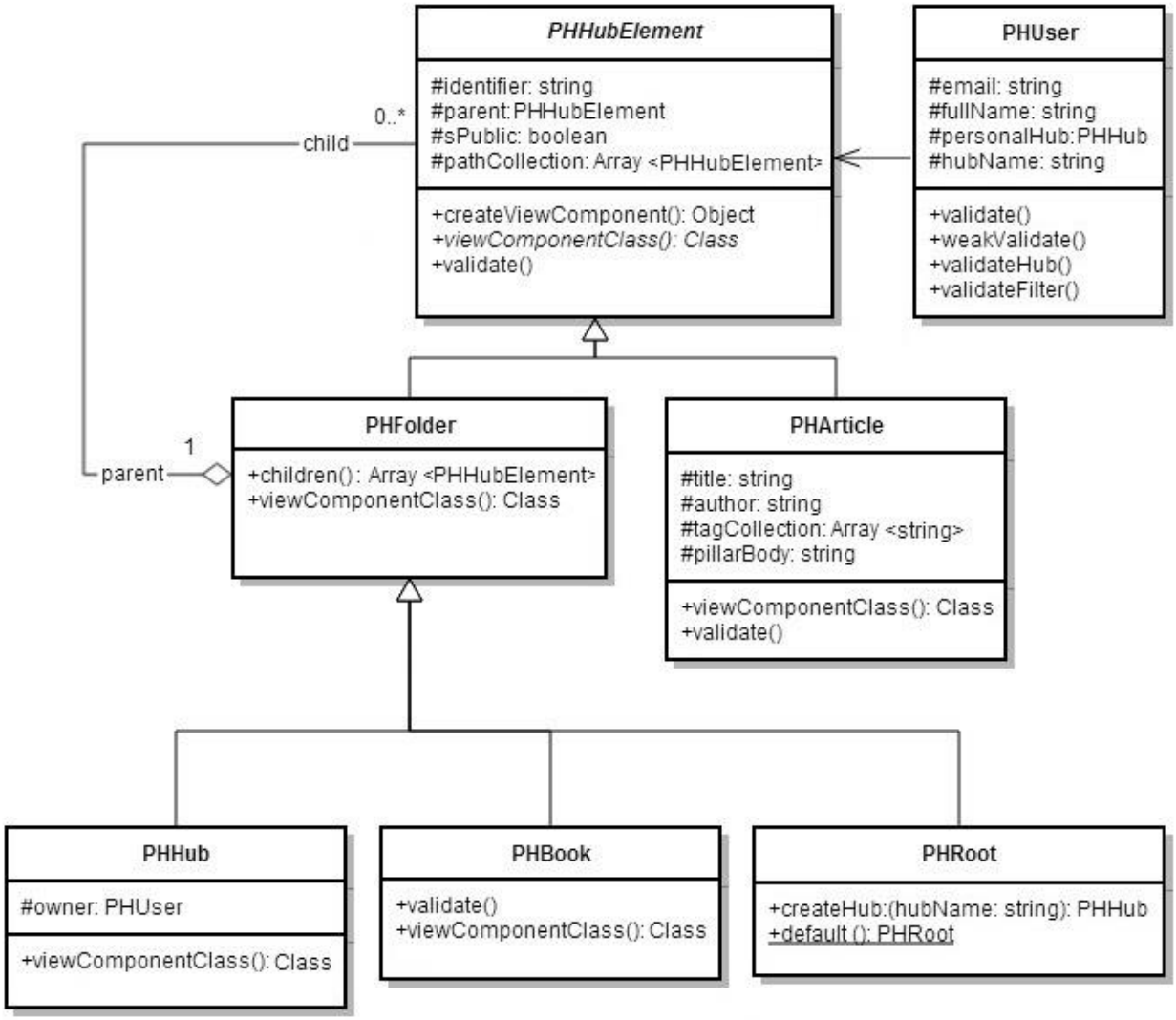

Рисунок 3.1 UML-диаграмма классов модели предметной области

Рассмотрим более подробно существующие классы модели:

Класс PHUser служит для хранении информации о пользователе. Для однозначной идентификации пользователя используется поле «email». «fullName» содержит в себе полное имя пользователя. Поле «personalHub» содержит ссылку на личный хаб пользователя, объект класса PHHub. Метод «validate» проверяет на валидность полное имя пользователя и название персонального хаба.

PHHubElement – абстрактный класс, семантически обозначающий любой элемент, содержащийся в модели, кроме PHUser. Содержит в себе 4

поля, присутствующих в каждом потомке. Поле «identifier» –это уникальный идентификатор, по сути – название элемента. Поле «isPublic» несет в себе информацию о том, доступен документ для всеобщего просмотра или нет. В поле «parent» находится объект-родитель данного элемента. Например, для всех хабов в системе родителем будет являться объект PHRoot. «pathCollection» – это список объектов-родителей, необходимый при навигации по документам.

PHHubElement имеет абстрактный метод «viewComponentClass». Для каждого класса-наследника этот метод возвращает соответствующее имя класса, отвечающего за отрисовку данного элемента. В зависимости от возвращаемого значения метода «viewComponentClass», методом «createViewComponent» будут создаваться компоненты необходимого подкласса. Такая структура позволяет выводить разнородные объекты с помощью одного интерфейса.

Класс PHArticle служит для хранении информации о статье. В поле «author» содержит имя автора статьи, в «title» лежит заголовок статьи. «tagCollection» представляет собой коллекцию тэгов, описывающих тематику данной статьи. Дата публикации задается в поле «publicationDate». Непосредственно тело статьи, текст на языке Pillar находится в «pillarBody». При отображении статьи на главное странице удобно ознакомиться с кратким ее содержанием; для этого создано поле «abstract», содержащее краткое описание статьи.

Класс PHFolder служит для хранении информации о папке как о некотором составном объекте. В поле «children» находится коллекция объектов типа PHHubElement, содержащихся в данной папке.

PHHub наследуется от PHFolder и представляет собой корневое хранилище документов для каждого пользователя. Помимо прочего, хранит в

поле «owner» ссылку на пользователя - создателя хаба.

PHBook наследуется от PHFolder и представляет собой упорядоченную коллекцию некоторого количества объектов-статей.

PHRoot наследник от Folder. Этот класс представляет собой корень файловой системы. Так как в системе в любой момент времени должен быть только один корневой объект, он реализуется с помощью паттерна Синглтон.

3.2 Реализация архитектуры.

Чтобы отразить основные значащие элементы приложения и в тоже время не перегрузить схему деталями, на ней представлены лишь модули приложения, разбитые по уровням в соответствии со схемой, описанной в пункте 2.2.3 данной работы. Результат можно наблюдать на рис. 3.2.

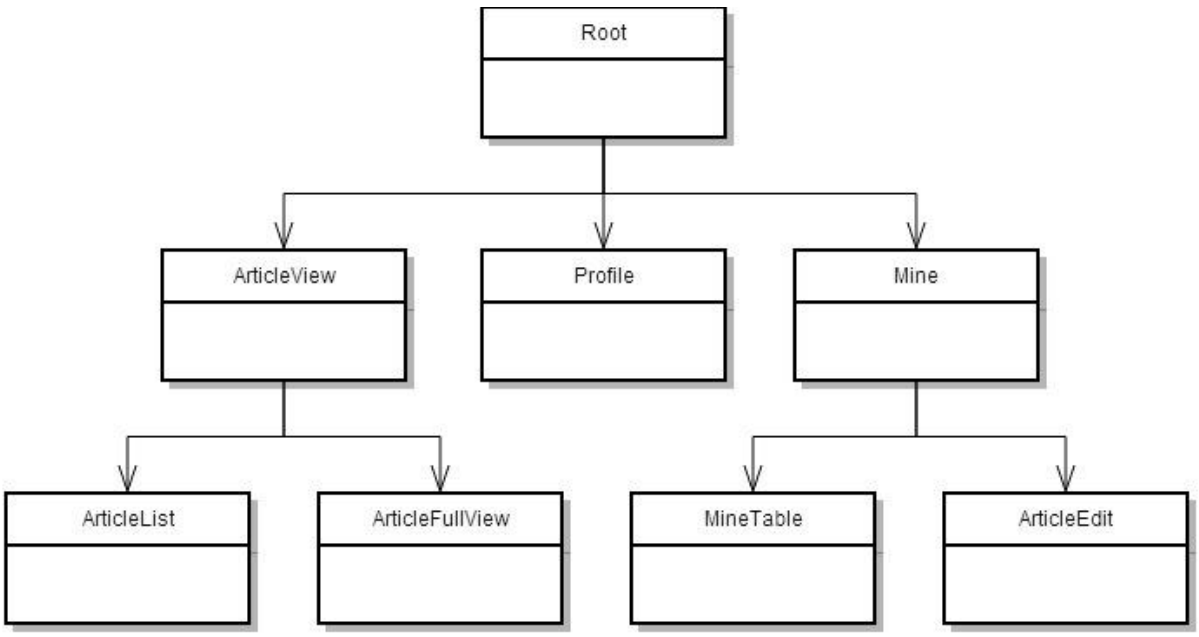

Рисунок 3.2 Упрощенная схема архитектуры

На первом уровне расположен Модуль Root. Он состоит из четырех классов: PHRootController, PHRootLayout, PHPublicHeaderComponent, PHPrivateHeaderComponent (см. рис. 3.3).

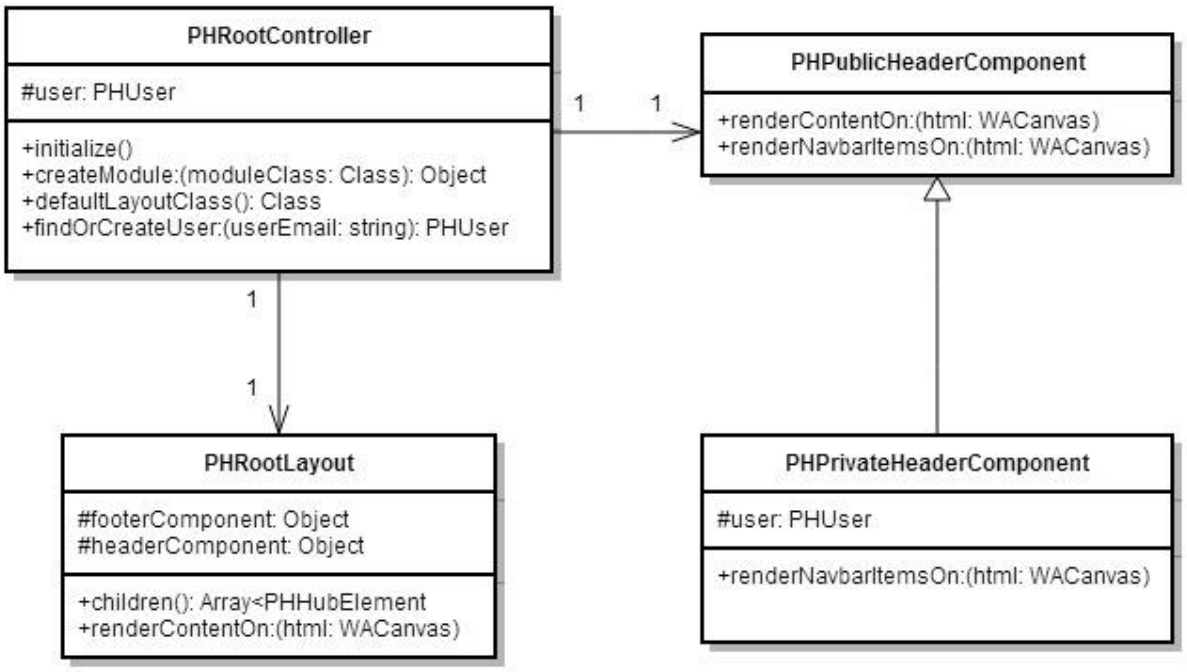

Рисунок 3.3 Диаграмма классов модуля Root

PHPublicHeaderComponent отображает хедер страницы в таком виде, в каком его видит неавторизованный пользователь.

PHPrivateHeaderComponent отображает хедер страницы в таком виде, в каком его видит авторизованный пользователь. Основное отличие в появлении дополнительных контролов, обеспечивающих возможность перейти на страницу с личными документами или зайти в личный профиль.

PHRootLayout имеет два поля – headerComponent и mainComponent. В headerComponent по умолчанию находится экземпляр класса PHPublicHeaderComponent. mainComponent предназначен для отображения какого-то конкретного компонента второго уровня.

PHRootController выполняет несколько задач:

- Обрабатывает аннаунсменты перехода между страницами, присылаемые ему модулями нижнего уровня. В результате обработки инициализируется один из модулей нижнего уровня, который подменяет собой текущее содержимое PHRootLayout’a.

- Обрабатывает MPUserLoginAttemptAnnouncement, который вызывается при попытке пользователя авторизоваться. В обработчике этого аннаунсмента проводится проверка – существует ли пользователь в базе данных. Если да, то система авторизует пользователя, если нет, то инициализируется форма PHNewProfileComponent, предназначенная для ввода данных, необходимых для работы пользователя с системой. В случае успешной авторизации контроллер присваивает полю headerComponent класса PHRootLayout экземпляр класса PHPrivateHeaderComponent.

На втором уровне расположены следующие модули:

- Profile модуль содержит классы для управления данными пользователя.

- ArticleView – модуль «Главная страница». Содержит в себе классы, необходимые для отображения списка опубликованных статей с возможностью фильтрации.

- Mine модуль. Он содержит в себе классы, необходимые для отображения страницы управления документами.

Модуль Profile состоит из двух компонентов – PHProfileComponent и PHNewProfileComponent (см. рис. 3.4).

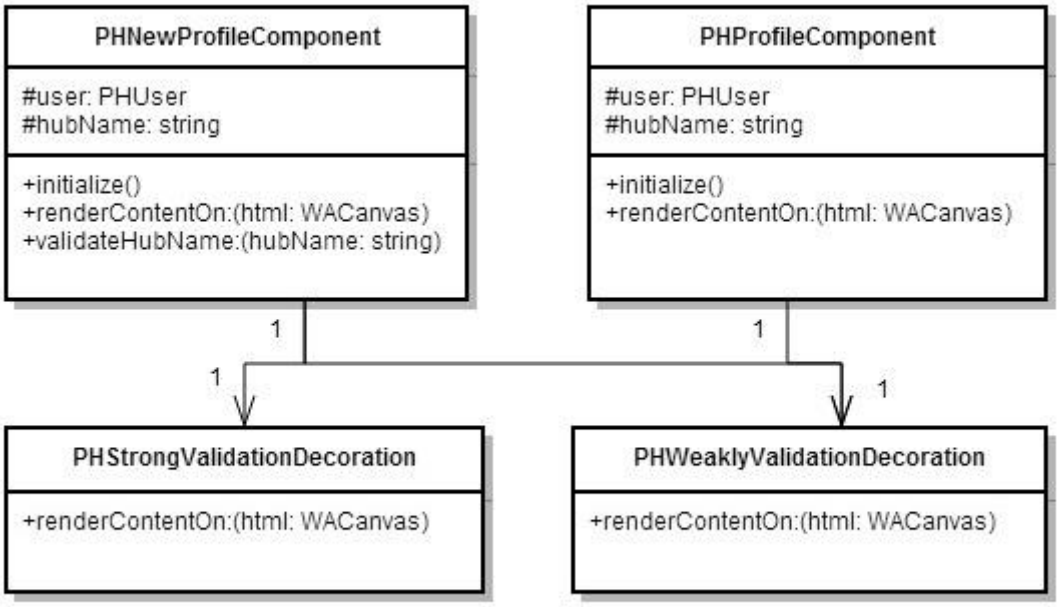

Рисунок 3.4 Диаграмма классов модуля Profile

PHProfileComponent предназначен для отображения данных о пользователе, таких как имя пользователя, емэйл, название персонального хаба. Пользователь может сменить лишь свое имя, остальные два поля статичны.

PHNewProfileComponent – по сути форма, отображающаяся при регистрации, для ввода персональных данных.

Так же в модуле Profile есть PHStrongValidationDecoration и PHWeakValidationDecoration. Они переопределяют стандартный способ отображения ошибок, которые возникают при вводе персональных данных.

Модуль ArticleView представляет из себя главную страницу веб-приложения и состоит из контроллера и двух модулей третьего уровня – ArticleList и ArticleFullView (см. рис. 3.5).

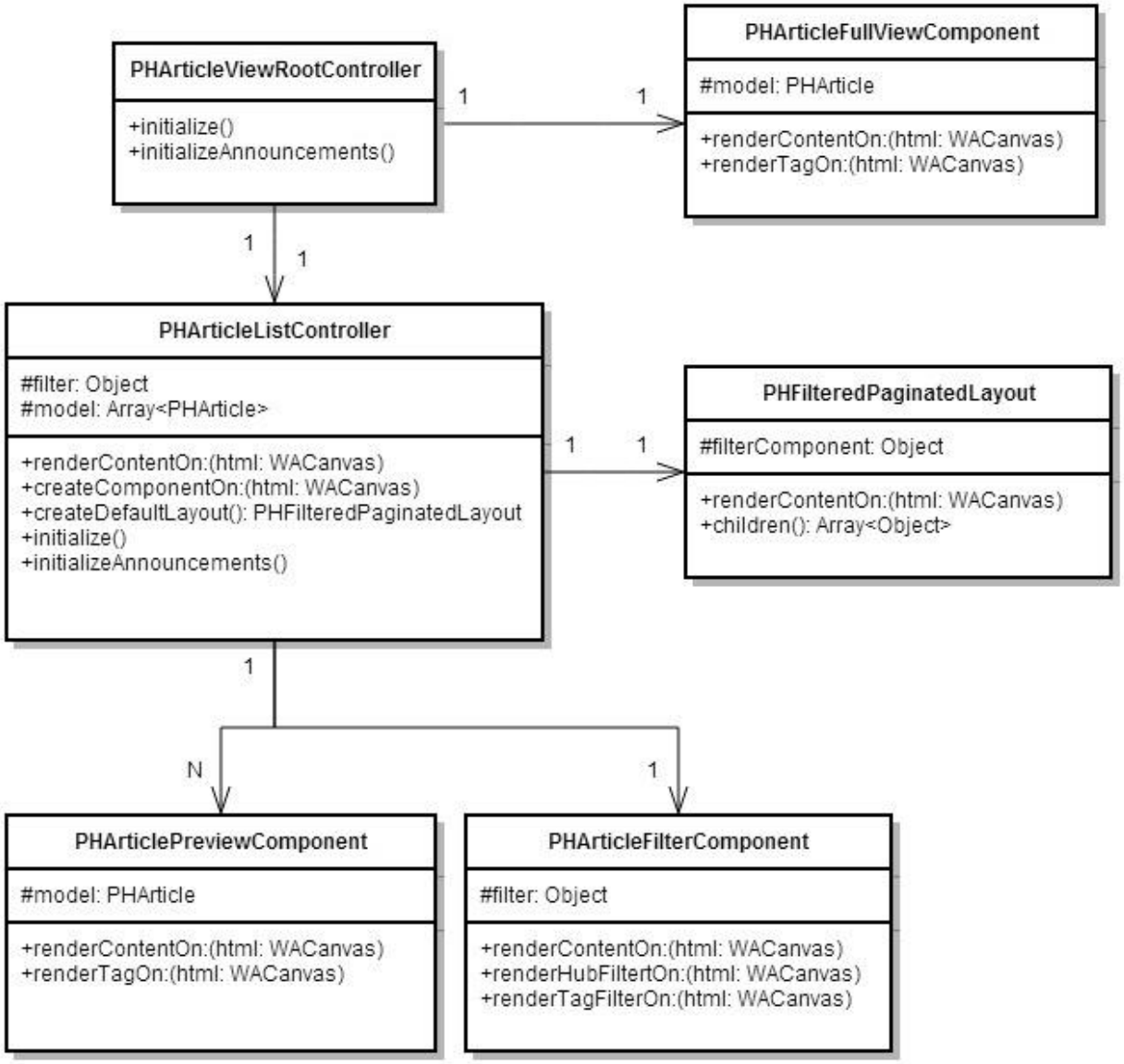

Рисунок 3.5 Диаграмма классов модуля ArticleView

Класс PHArticleViewRootController является контроллером модуля ArticleView. Его обязанность заключается в обработке аннаунсментов для перехода между страницами, присылаемые ему модулями нижнего уровня. В результате обработки инициализируется и отображается один из модулей нижнего уровня. В частности, при получении PHViewArticleClickedAnnouncment контроллер запустит режим просмотра документа путем создания экземпляра класса PHArticleFullViewComponent.

Класс PHArticleFullViewComponent предназначен для отображения какой-либо конкретной опубликованной статьи в режиме чтения. При просмотре текст статьи отображается уже не на языке Pillar, а в виде HTML.

Модуль PHArticleList предназначен для отображения списка документов, доступных для просмотра.

Класс PHArticleListController содержит поле model, в котором находится коллекция опубликованных статей. Каждая из этих статей отображается с помощью PHArticlePreviewComponent. В обязанности этого компонента входит отображение краткой информации о статье, такой как название, краткое содержание, дата публикации, автор, хаб создателя статьи, тэги.

Так как статей может быть очень много, для удобства просмотра была организована пагинация (разбиение информации на страницы). Класс PHFilteredPaginatedLayout отвечает за размещение компонентов, содержащихся в модуле PHArticleList.

PHArticleFilterComponent предназначен для отображения фильтров. С помощью метода renderHubFilterOn: отображается выпадающее меню, значение которого по умолчанию равно «All». При значении фильтра «All» отображаются все статьи, то есть фильтрация не производится.

Чтобы отфильтровать статьи, необходимо кликнуть по названию хаба в кратком описании статьи. Тогда сообщение о смене хаба PHFilterHubChangedAnnouncment с аргументом hub пошлется контроллеру модуля ArticleList. Этот контроллер отобразит только те элементы, которые принадлежат выбранному хабу.

Модуль Mine состоит из двух модулей третьего уровня – PHMineTable и PHArticleEdit (см. рис. 3.6)

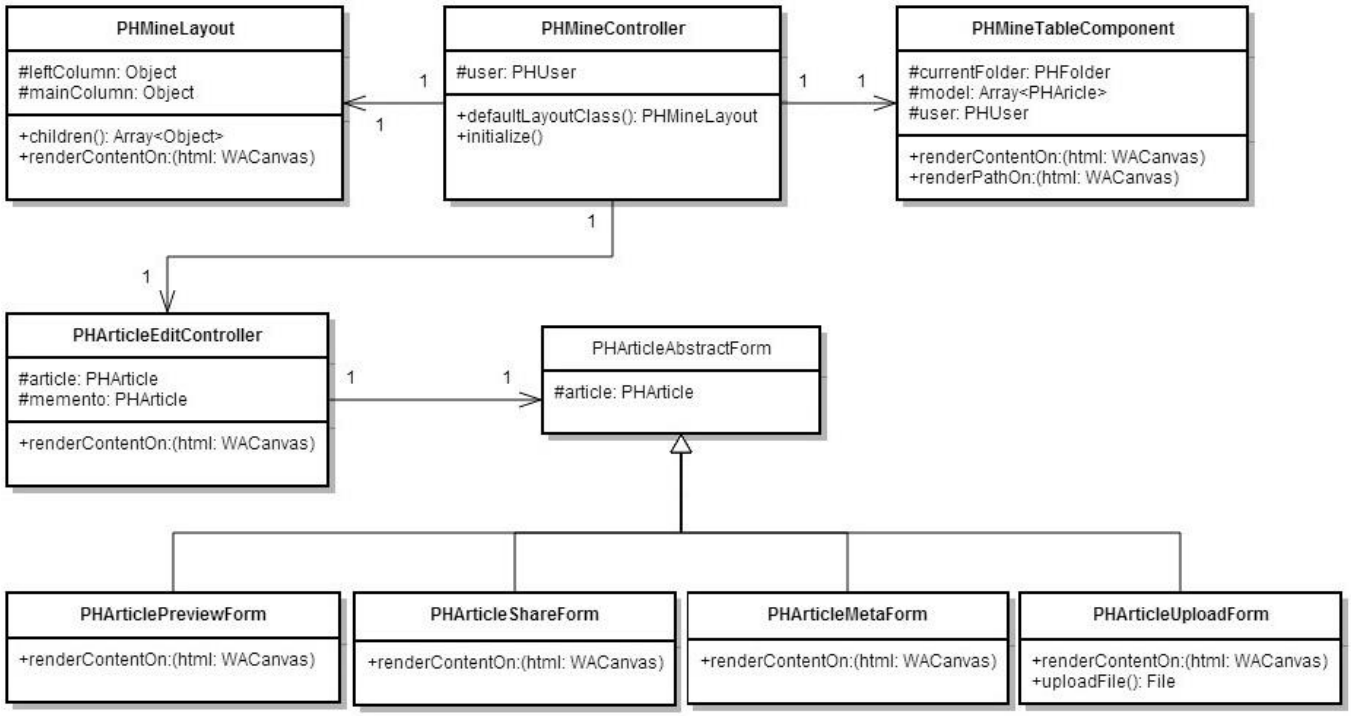

Рисунок 3.6 Диаграмма классов модуля Mine

Класс PHMineController обрабатывает аннаусменты перехода между страницами, присылаемые ему модулями нижнего уровня. В результате обработки инициализируется и отображается один из модулей нижнего уровня.

Модуль PHMineTable состоит из одного класса – PHMineTableComponent. Этот класс предназначен для отображения списка документов, содержащихся в хабе текущего пользователя. С помощью поля currentFolder, хранящего в себе текущую папку – объект классаPHFolder, и метода pathCollection реализована навигация по каталогу хаба.

При создании новой статьи или редактировании уже существующий PHMineTableComponent пошлет аннаусмент PHMineController’у, который отобразит модуль PHArticleEdit.

Модуль PHArticleEdit состоит из контроллера и нескольких форм. PHAricleEditController предназначен для управления формами редактирования документа. Так же контроллер хранит два состояния статьи,

исходное, в поле article, и текущее, в поле memento. Если в статью были внесены какие-либо изменения, они отобразятся на ее текущем состоянии. При несовпадении значений полей article и memento контроллер просигнализирует пользователю о том, что имеются несохраненные данные.

Форма PHArticleMetaForm предоставляет интерфейс для заполнения или редактирования метаданных о статье.

Форма PHArticlePreviewForm предоставляет пользователю возможность просмотреть свою статью, преобразованную в HTML.

Форма PHArticleShareForm предоставляет возможность пользователю сделать его статью опубликованной, изменяя значения поля isPublic.

Форма PHArticleUploadForm предоставляет интерфейс для загрузки файла на языка Pillar с компьютера пользователя на сервер.

Таким образом, с помощью создания вышеперечисленных классов архитектура приложения была реализована в полной мере.

3.3 Оформление интерфейса.

Интерфейс для приложения разрабатывался в пользу удобства для пользователя, а не внешней привлекательности. Однако в своем приложении для придания ему вида готового продукта был использован Bootstrap for Seaside[13]. Это свободный набор инструментов для создания веб-приложений, включающий в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов.

Цветовая схема проекта так же выдержана в строгом, однообразном стиле, чтобы пользователь не отвлекался от своих задач.

Конкретные примеры интерфейса представлены в приложении А. Руководство пользователя.

Заключение

В рамках данной работы была спроектирована модель данных для предметной области, на основе этой модели была создана база данных. Была спроектирована общая архитектура приложения. Так же были спроектированы и реализованы основные модули веб-приложения. Кроме этого, была проведена работа по визуальному оформлению приложения.

Таким образом, все поставленные задачи выполнены, и цель работы достигнута.

Результатом проделанной работы стало веб-приложение, в котором реализованы необходимые инструменты для просмотра, создания и редактирования статей и проектов на языке Pillar.

В настоящее время заканчивается доработка приложения, добавляются дополнительные модули, производится тестирование. Планируется публикация приложения в сети Интернет на open-source основе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. D.Cassou Pillar-documentation: [Электронный ресурс] // GitHub, inc 2014, URL: https://github.com/pillar-markup/pillar-documentation (дата обращения: 24.02.2014)

2. PierCMS [Электронный ресурс] // URL: http://www.piercms.com/doc (дата обращения: 9.03.2014)

3. WebAnnouncment [Электронный ресурс] // URL: http://smalltalkhub.com/#!/~mikefilonov/WebAnnouncement (дата обращения: 14.03.2014)

4. A. P. Black, S. Ducasse, O. Nierstrasz, D. Pollet Pharo by Example - Square Bracket Associates, 2009

5. Pharo [Электронный ресурс] // URL: http://pharo.org/ (дата обращения: 3.11.2013)

6. S. Ducasse, L. Renggli, D. C. Shaffer, R. Zaccone Dynamic Web Development with Seaside - Square Bracket Associates, 2009

7. Seaside [Электронный ресурс] // URL: http://seaside.st (дата обращения: 3.11.2013)

8. E. Gamma Design patterns: elements of reusable object-oriented software - Addison-Wesley Longman Publishing Co., Inc, 2007

9. S. Alpert, K. Brown, B. Woolf The Design Patterns Smalltalk Companion: [Электронный ресурс] // URL: http://stephane.ducasse.free.fr/FreeBooks/SmalltalkDesignPatternCompanion/ (дата обращения: 8.04.2014)

10. SandstoneDB [Электронный ресурс] // squeakSource, 2014

URL: http://www.squeaksource.com/@B1qFurkm9jCzuqE_/hr1PHtx_

(дата обращения: 10.03.2014)

11. R. Leon SandstoneDb, Simple ActiveRecord Style Persistence in Squeak: [Электронный ресурс] // URL: http://onsmalltalk.com/sandstonedb-simple-activerecord-style-persistence-in-squeak (дата обращения: 2.04.2014)

12. Mozilla Persona [Электронный ресурс] // Mozilla Developer Network, 2014 URL: https://developer.mozilla.org/en-US/Persona (дата обращения: 29.04.2014)

13. Bootstrap for Seaside [Электронный ресурс] // URL: http://pharo.pharocloud.com/bootstrap (дата обращения: 11.05.2014)

Приложение А. Руководство пользователя.

1. Системные требования.

Для использования веб-приложения необходима рабочая станция с наличием Интернет-соединения со скоростью не менее 256 килобит в секунду и установленным на ней одним из веб-браузеров:

- Google Chrome 8 и выше;
- Microsoft Internet Explorer 7 и выше;
- Apple Safari 5 и выше;
- Mozilla Firefox 3.6 и выше;
- Opera 10 и выше.

Для корректной работы сайта не требуется установка дополнительных приложений или плагинов. Установка Flash-плеера может потребоваться исключительно для отображения некоторых баннеров рекламодателей, и не затрагивает остальные разделы и элементы сайта.

Кроме того, приложению желательно разрешение на сохранение cookies на компьютере пользователя. В частности, это позволит сделать более удобной процедуру авторизации при повторных визитах на сайт.

Эксплуатация системы на мобильных платформах возможна, однако достаточная функциональность и работоспособность не гарантируется.

2. Авторизация на сайте.

Для того чтобы полноценно использовать весь функционал сайта, необходимо стать пользователем этой системы, то есть авторизоваться. Для этого необходимо нажать кнопку «Sign in» в верхней части сайта. Откроется форма авторизации Mozilla Persona[12]. В системе Persona нет такого понятия, как логин для входа на сайт, пользователь идентифицируется при

помощи одного из своих адресов электронной почты. Необходимо заполнить поля для ввода e-mail’а и пароля от этого e-mail’а.

После ввода данных и нажатия кнопки «Войти» данные, введенные в эти поля, отправятся проверяющей стороне. Если данные прошли проверку, и если это первая авторизация этого пользователя в рамках данного веб-приложения, ему будет предложено заполнить необходимую для дальнейшей работы информацию.

В поле «Your Name» необходимо ввести полное имя или псевдоним. Это имя, как правило, будет отображаться в качестве автора статьи.

В поле «Hub identifier» необходимо внести название пользовательского хаба (хаб – это вместилище всех будущих работ). Пользователь должен ответственно подойти к этому моменту, потому что название для хаба дается раз и навсегда, изменять его потом нельзя. После ввода всех данных и нажатия кнопки «Save» будет произведена проверка корректности введенных данных. Если пользователь оставил поля пустыми, или ввел синтаксически неприемлемую конструкцию, или название для хаба уже используется другим пользователем – появится уведомление, содержащее описание возникшей ошибки. Если все верно, информация о пользователе сохранится, и он окажется на главной странице сайта в залогиненном состоянии.

3. Просмотр документов

На главной странице сайта отображается список статей и проектов, доступных для просмотра (см. рис. А.1).

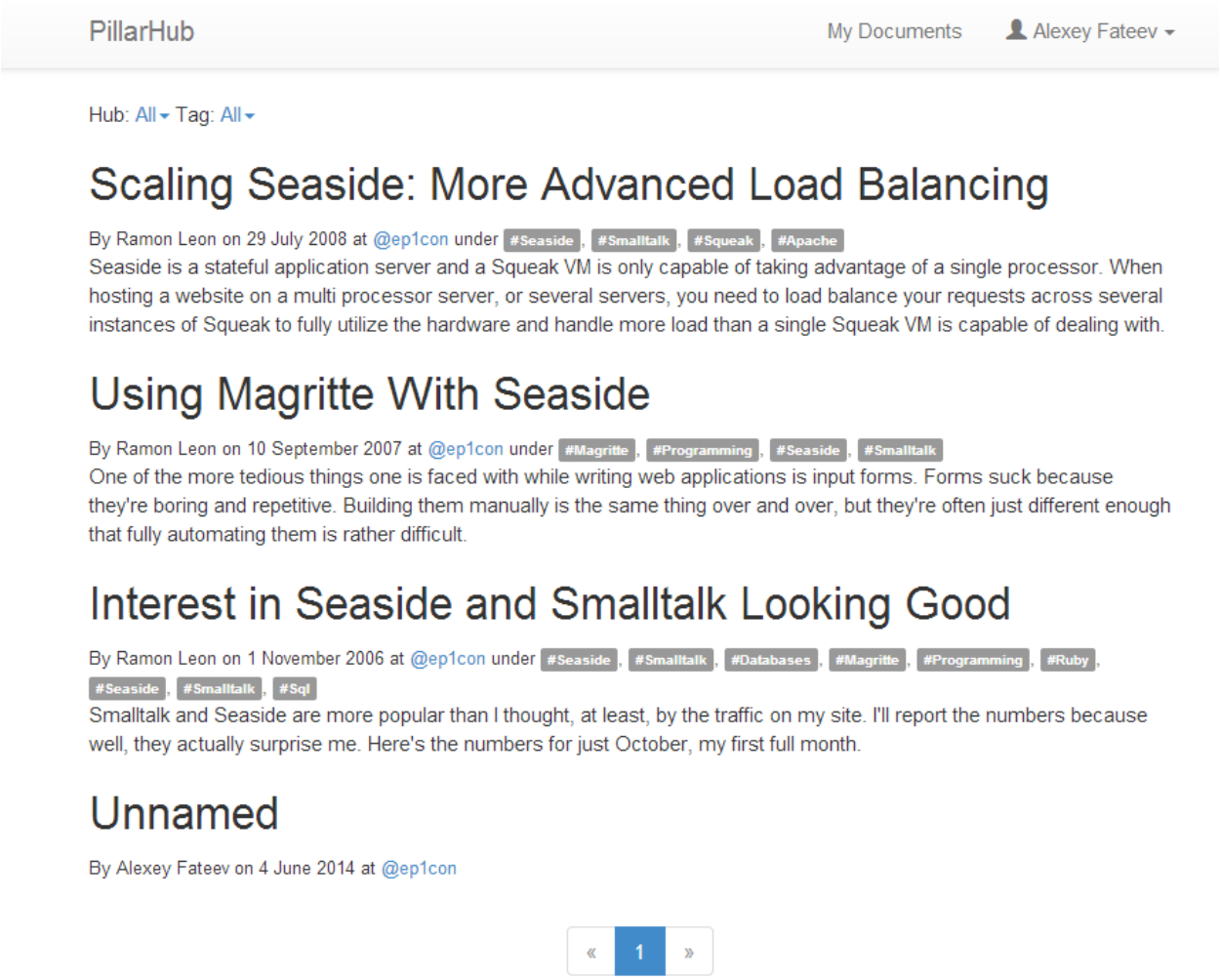

Рисунок А.1 Главная страница

В верхней части страницы находятся три выпадающих меню: в меню «Hub» можно выбрать конкретный хаб, тогда отобразятся лишь принадлежащие этому хабу статьи и проекты. Второе выпадающее меню «Tag» позволяет осуществить поиск документов по тегу.

На главной странице отображается лишь информация о документе; чтобы посмотреть содержимое, необходимо кликнуть по названию документа. При этом откроется страница просмотра документа. Обратно на главную страницу из режима просмотра можно перейти, нажав кнопку «Back».

4. Управление документами.

Чтобы создавать, редактировать или удалять документы, необходимо перейти по ссылке «My Documents», расположенной вверху страницы. На открывшейся странице расположен список всех созданных пользователем документов и папок (см. рис. А.2). В заголовке отображается путь до текущей папки.

Рисунок А.2 Список документов для редактирования

Список элементов представлен в виде таблицы, где каждая строка – это либо статья, либо проект, либо папка. Первая ячейка каждой строки может принимать вид одной из двух пиктограм:

- статья опубликована

- статья неопубликована

Вторая ячейка каждой строки в зависимости от типа элемента может принимать вид одной из трех пиктограмм:

- строка представляет собой статью

 - строка представляет собой проект

 - строка представляет собой папку

В третьей ячейке находится название элемента. Если элемент - это статья - щелчок по названию вызовет форму редактирования документа. Если элемент – это папка или проект, щелчок по названию отобразит таблицу с подкаталогами этого элемента.

В четвертой ячейке отображается дата публикации статьи.

В пятой ячейке отображается автор статьи.

В шестой ячейке отображается пиктограмма  . Нажав на нее, можно вызвать форму редактирования элемента.

В последней ячейке находится пиктограмма  . Нажав на нее, можно удалить статью.

Для того чтобы создать свою статью на языке Pillar, необходимо нажать на кнопку «New Article» - отобразится форма для создания документа. На ней расположено поле для ввода текста, а в заголовке находятся 5 кнопок со следующим функционалом (см. рис. А.3):

- «Save» – сохраняет статью. Как только пользователь изменяет тело статьи или метаинформацию о ней, надпись на иконке становится «Save*», что означает, что имеются несохраненные данные, которые необходимо сохранить.

- «Import» – отображает форму, позволяющую загружать Pillar-документы, хранящиеся на вашем компьютере.

- «Meta» – при нажатии на эту кнопку откроется форма для ввода метаданных о статье, таких как заголовок, автор, дата публикации, список

тэгов, краткое содержание. Поля «Author» и «Date» заполняются автоматически, но при желании пользователь может изменить их.

- «Preview» – позволяет посмотреть статью в конечном виде, как она будет выглядеть в публикации

- «Share» – при нажатии на эту кнопку отобразится форма, где пользователь может, кликнув по чекбоксу «Public Access», опубликовать свою статью на главной странице сайта. Также пользователь может задать «Document URL» – постоянную ссылку на статью. Эту ссылку удобно использовать, когда необходимо прорекламировать свой труд за пределами данного веб-приложения.

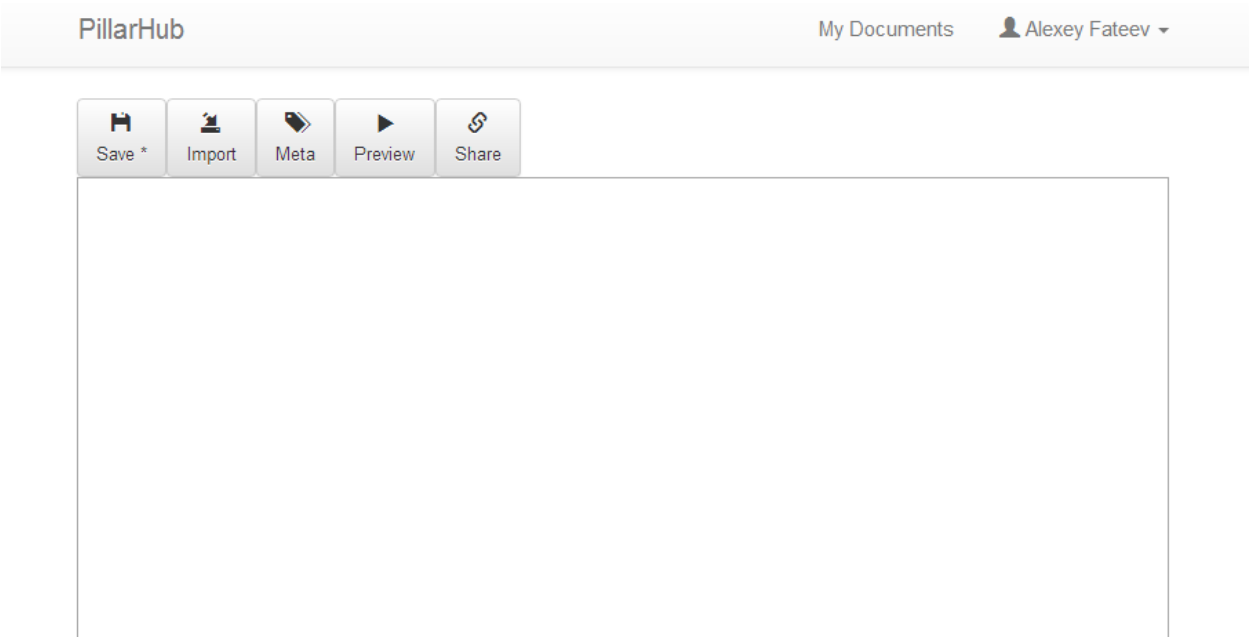

Рисунок А.3 Форма для создания документа

В верхнем правом углу расположено выпадающее меню с именем пользователя. Нажав на него, можно выбрать пункт «Edit Profile». Откроется страница с личной информацией (см. рис. А.4). Чтобы разлогиниться, достаточно нажать кнопку «Sign out», расположенную в выпадающем меню.

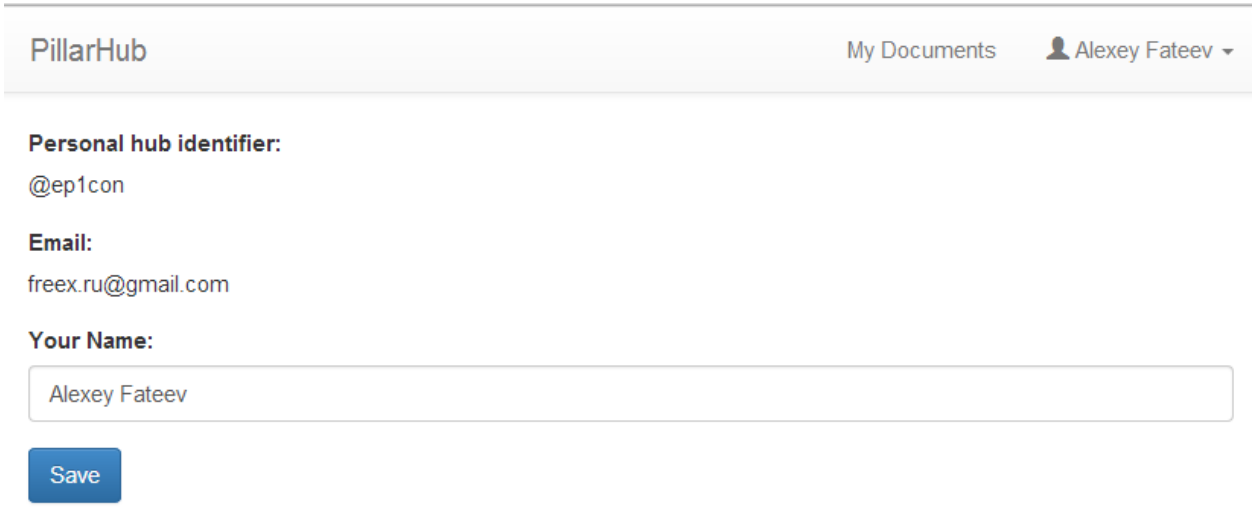

Рисунок А.4 Страница личной информации