# Разработка и применение SaaS-приложения конкурентного анализа для ЛПР от производителей ОАТ

**В третьей главе** описана разработка Web-приложения конкурентного анализа для производителя объектов авиационной техники на основании представления конкуренции в данном сегменте рынка как ССС (первая глава) и математических моделей и алгоритмов, разработанных во второй главе.

## Разработка функциональной модели конкурентного анализа ОАТ

Первая часть главы посвящена разработке функциональной модели конкурентного анализа для описания бизнес-процессов использования данного SaaS-приложения топ-менеджерами предприятий - производителей объектов боевой авиации. Для этого была выбрана методология IDEF0, позволяющая сделать акцент на взаимосвязи процессов. Бизнес-процесс представляется в детализированном виде, позволяющем эффективно взаимодействовать системным аналитикам, программным архитекторам, специалистам в предметной области и другим занятым в проекте лицам. На верхнем уровне модели (рис. 33, диаграмма A0) определяются основные входы (Input, I1-I3) и выходы (Output, O1-O3), управляющие воздействия (Control, C1-C3) и исполнители (M1-M3).

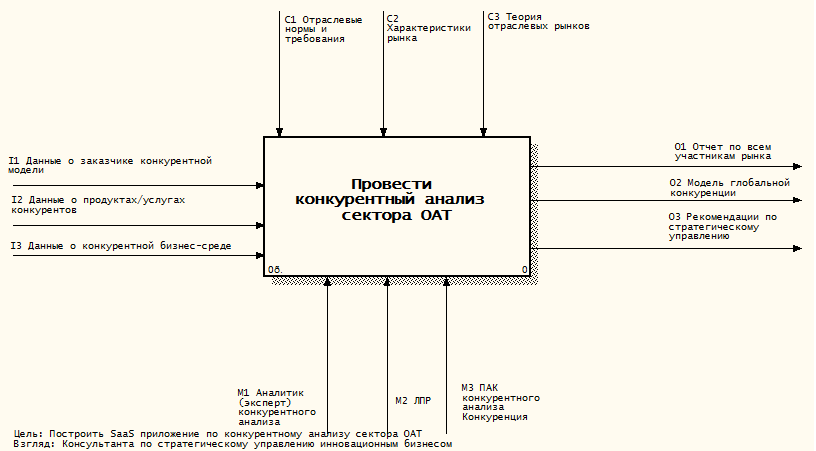


Рис.33. Функциональная модель конкурентного анализа

сектора ОАТ в виде IDEF0-диаграммы (A-0)

Декомпозиция диаграммы A0 отображает основные этапы конкурентного анализа (рис. 34) в интересах заказчика – ЛПР от производителя ЛА:

1. проведение анализа данных о заказчике и его продукте;
2. анализ данных конкурентной среды рынка или сегмента рынка заказчика;
3. выявление параметров бизнеса основных конкурентов для использования в агентной модели;
4. построение модели глобальной конкуренции на рынке заказчика;
5. проведение вычислительного эксперимента с целью выработки рекомендаций по формированию конкурентной стратегии заказчика.

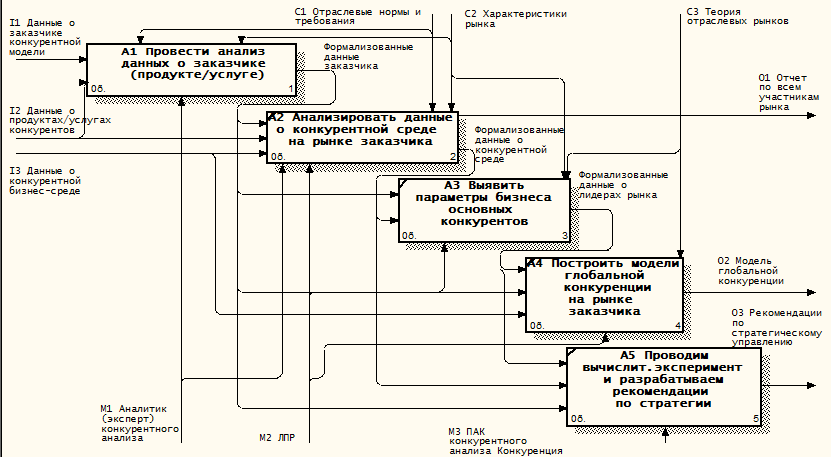


Рис. 34. Декомпозиция процесса конкурентного анализа (A0)

Рассмотрим декомпозицию четвертого и основного из перечисленных выше этапов конкурентного анализа (A4). Для построения модели глобальной конкуренции на рынке заказчика (рис. 35) требуется:

1. построить фрактальный нагруженный граф для сегмента или рынка заказчика;
2. определить все виды отношений между агентами и зафиксировать в виде таблицы;
3. задать и детализировать алгоритмы поведения агентов рынка;
4. построить на основании предыдущих этапов модель глобальной конкуренции.

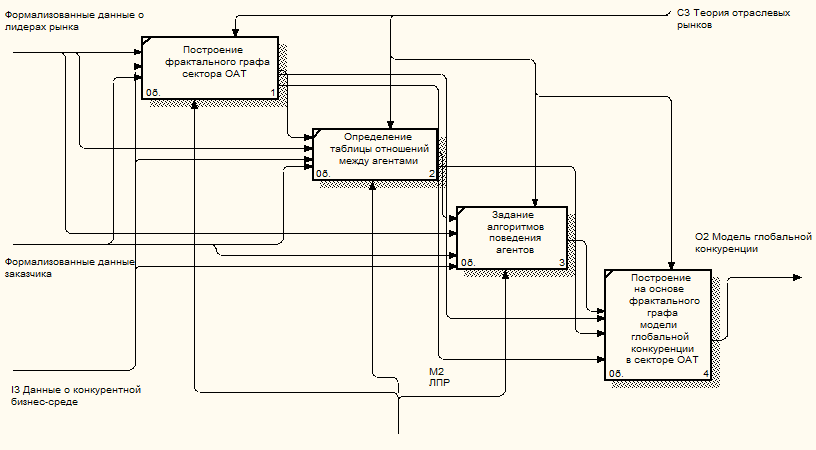


Рис. 35. Декомпозиция процесса построения модели глобальной конкуренции

на рынке заказчика

## Выбор средств разработки программного комплекса «Сompetition»

### Концепция MVC

MVC (Model View Controller/Модель - Вид (Представление) -Контроллер) – это проверенная и надежная схема, дающая возможность быстро разработать модульное и расширяемое приложение. Разбивка приложения на отдельные блоки (виды, модели и контроллеры) позволяет вести разработку прямо "на ходу". Новые возможности легко могут быть добавлены, а старые изменены. Модульная структура приложения позволяет разработчику и дизайнеру работать одновременно. Существует, кроме того, возможность быстрого прототипирования. Разделение на модули позволяет изменять часть программы, не затрагивая остальные модули[[1]](#footnote-2). Стандартная схема архитектуры[[2]](#footnote-3) «Модель-Вид-Контроллер» изображена на рис. 36:

**Представление**

**Модель**

**Контроллер**

1. Обновление представления
2. Оповещение об изменениях
3. Модификация
4. Действие

Рис. 36. Схема разработки ПО MVC

Рассмотрим данную схему подробнее. Шаблон MVC содержит в себе три основных компонента:

* *Модель* – отвечает за предоставление данных и методов работы с ними, а также реагирует на запросы, изменяя свое состояние. Модель отображает внутреннее устройство системы.
* *Вид (Представление)* **–** отвечает за отображение информации, поступающей из системы или в систему.
* *Контроллер –* обеспечивает связь между пользователем и системой. Он получает данные от пользователя (через Вид) и передает их в Модель, и наоборот, получает сообщения от Модели и передает их в Вид.

Еще один вариант схемы MVC[[3]](#footnote-4) предлагают разработчики CakePHP.

****

Рис. 37. Схема MVC на примере фрэймворка CakePHP

На этой схеме (рис. 37) отображен процесс взаимодействия клиента с системой, состоящий из 6 последовательных действий:

1. Клиент активирует ссылку, после чего браузер делает запрос к серверу.
2. Диспетчер проверяет URL запроса и перенаправляет запрос соответствующему контроллеру для дальнейшей обработки.
3. Контроллер действует согласно своей логике и обращается к модели (как правило, к таблице БД, для извлечения, добавления, удаления или проверки информации)
4. Модель отвечает на запрос контроллера, обмениваясь с ним данными.
5. После обработки данных контроллер передает их Виду. Вид, получая данные, готовит их к отображению браузером.
6. Используя данные, полученные от контроллера, Вид передает клиенту полностью отформатированный ответ.

### Выбор СУБД для реализации БД

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных[[4]](#footnote-5).

Так как разрабатываемое нами SaaS-приложение конкурентного анализа имеет клиент-серверный тип архитектуры информационной системы, то при выборе СУБД мы будем ориентироваться на клиент-серверный способ доступа к БД. Клиент-серверная СУБД может рассматриваться как распределенная система, в которой все запросы создаются на одном узле (клиент), а вся обработка выполняется на другом (сервер), если считать для простоты, что имеется лишь один узел клиента и один узел сервера[[5]](#footnote-6).

Выделим основные СУБД, отвечающие нашим требованиям:

* Oracle
* PostgreSQL
* MySQL

Мы остановили свой выбор на СУБД MySQL по следующим причинам:

* На подавляющем большинстве хостингов по умолчанию имеется доступ к СУБД MySQL, чего нельзя сказать о PostgreSQL и Oracle (доступ к ним предоставляется, как правило, за отдельную плату).
* MySQL является наиболее распространенной СУБД, вследствие чего образует вокруг себя активное сообщество и развитую службу поддержки.
* До начала разработки SaaS-приложения конкурентного анализа наибольший опыт работы мы имели именно в среде MySQL.

### Особенности использования фреймворка CakePHP

CakePHP — это программный каркас (framework) для создания веб-приложений[[6]](#footnote-7), написанный на языке PHP и построенный на принципах открытого ПО. CakePHP реализует паттерн «Модель-Вид-Контроллер» (MVC).

Причин, по которым был выбран данный программный каркас, несколько:

* продолжительный опыт работы разработчика с CakePHP до начала разработки SaaS-приложения конкурентного анализа;
* активное сообщество, служба online-поддержки, а также команда разработчиков, постоянно развивающая и качественно тестирующая программный каркас;
* бесплатность и открытость программного кода ядра;
* MVC архитектура;
* быстрая и гибкая шаблонизация;
* простое добавление модулей;
* возможность локализации;
* гибкая система кэширования;
* проработанный механизм, обеспечивающий безопасность и сохранность данных;
* развитая система общения с БД (PostgreSQL, MySQL, SQLite, Oracle Database)[8];
* генерация программного кода по схеме базы данных;
* компоненты для генерации и заполнения форм;
* простое внедрение AJAX-технологии.

Важно отметить, что программирование на CakePHP базируется на особых соглашениях (наименование таблиц БД, файлов) и нотациях (например, lowerCamelCase[[7]](#footnote-8)). Может показаться, что подобные ограничения негативно сказываются на гибкости, однако на практике оказывается, что, при поддержке требуемых соглашений процесс разработки только ускоряется, а гибкость совершенно не страдает.

Как уже было сказано ранее, CakePHP следует паттерну разработки MVC, что позволяет разделить приложение на три основные части (рис. 38):

* Модель (Model) – включает в себя файлы из папки /models;
* Вид (View) – включает в себя файлы из папки /views;
* Контроллер (Controller) – включает в себя файлы из папки /controllers.

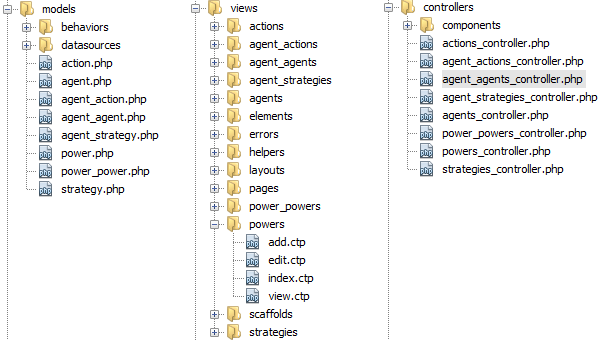


Рис. 38. Файловая структура SaaS-приложения

Хотелось бы особо подчеркнуть концептуальную сторону использования CakePHP. Для нас представляется важным непрерывность процесса разработки SaaS-приложения. Включаясь в этот процесс на завершающем этапе, CakePHP позволяет перевести схему базы данных (которая будет транслирована из Microsoft Visio, на основе диаграммы классов) в программный код при помощи утилиты bake[[8]](#footnote-9). Таким образом, мы моментально получаем набор файлов: models, controllers, views, которые образуют каркас SaaS-приложения. На рисунке 39 представлен пример сгенерированного контроллера для сущности Power.

Еще одной важной особенностью использования CakePHP является упрощенное взаимодействие с базой данных. Дело в том, что стандартная функция PHP, посылающая запрос MySQL, возвращает дескриптор результата запроса[[9]](#footnote-10), который в дальнейшем нужно передавать в функцию, работающую с результатами запросов, которые в свою очередь возвращают данные не в самом удобном формате. CakePHP предлагает готовые решения для получения информации (find, query[[10]](#footnote-11)), сохранения информации (save, saveAll[[11]](#footnote-12)) и удаления информации (delete[[12]](#footnote-13)) из БД. Эти решения оперируют массивами и имеет возможность гибкой настройки для получения данных в наиболее подходящей форме.



Рис. 39. Пример генерации программного кода с помощью утилиты bake

## Разработка высокоуровневой архитектуры программного комплекса «Competition» конкурентного анализа

### Особенности использования модели SaaS

SaaS (Software as a Service) представляет собой современную модель продажи программного обеспечения (ПО), при которой поставщик разрабатывает web-приложение и самостоятельно управляет им, предоставляя заказчикам доступ к ПО через Интернет. В отличие от модели SaaP (Software as a Product) в модели SaaS:

* доступ к ПО предоставляется удаленно по сетевым каналам через web-интерфейс;
* ПО развертывается в едином дата-центре (или дата-центрах), а не на мощностях каждого конкретного заказчика;
* ПО предоставляется на условиях аренды, и за него взимаются периодические платежи (стоимость технической поддержки обычно включается в стоимость арендной платы);
* обновления ПО устанавливаются централизованно на стороне провайдера.

Таким образом, модель SaaS обладает рядом существенных преимуществ как для пользователя ПО, так и для поставщика/разработчика (см. табл. 6).

Преимущества модели SaaS перед SaaP. Таблица 6

|  |  |
| --- | --- |
| **Для пользователя** | **Для поставщика/разработчика** |
| * Отсутствие затрат, связанных с установкой, обновлением и поддержкой работоспособности программного обеспечения; * Отсутствие аппаратной платформы, связанной с развертыванием ПО; * Прозрачный подсчет TCO (*TotalCostofOwnership*– совокупная стоимость владения); | * Борьба с пиратством; * Уменьшение затрат на развертывание и внедрение ИС (за счет развертывания и развития на «родной» аппаратной среде); * Рост популярности web-сервисов среди конечных пользователей; * Легкое проникновение на глобальные рынки; |

Продолжение таблицы 6.

|  |  |
| --- | --- |
| **Для пользователя** | **Для поставщика/разработчика** |
| * Отсутствие забот о своевременном обновлении ПО; * Быстрота внедрения; * Понятный интерфейс; * Возможность получить более высокий уровень обслуживания ПО. | * Бурное развитие web-технологий, большие функциональные возможности web-приложений, простота их реализации; * Привязка заказчика (заказчик не может отказаться от услуг поставщика и продолжать использовать систему); * В долгосрочном периоде доходы от продаж SaaS выше, чем от SaaP |

Однако стоит признать, что SaaS при всех своих преимуществах обладает следующими недостатками:

* клиент привязывается к единственному поставщику/разработчику, что может быть для него (клиента) нежелательно;
* возникают вопросы безопасности и утечки информации (решаются на 3-х уровнях: юридическом, технологическом, информационном);
* требуется постоянное подключение клиента к Интернету;
* универсальные SaaS-решения могут не работать для компаний со специфическими потребностями[[13]](#footnote-14).

При всех перечисленных выше недостатках преимущества модели SaaS весьма весомы. Уже к 2011 году 25% всех ИТ-решений для бизнеса в мире будут поставляться согласно этой модели[[14]](#footnote-15). Таким образом, 1/3 всех расходов на ИТ будут тратиться компаниями на web-решения.

На российском рынке бизнес-ИТ модель SaaS уже предоставлена разработками компаний SugarCRMInc[[15]](#footnote-16) (*SugarCRM*), NetSuiteInc (*NetSuiteERP*), NetSuiteInc (*NetSuiteCRM*), Веспол (*BomaAnalyticOnline*), Логнекс (*МойСклад*), Альтсофт (*Кларис*), Мегаплан (*Мегаплан*), БОСС. Кадровые системы (БОСС-Кадровик).

Появляются специализированные SaaS-решения для поддержки процесса стратегического управления. Компания QPRSofrwarePlc уже сейчас предоставляет SaaS-продукт QPRScoreCard, который позволяет моделировать систему сбалансированных показателей, распространяя понимание стратегии и динамики ее исполнения на всех организационных уровнях.

Список основных SaaS BI[[16]](#footnote-17)-решений представлен в таблице 7.

Обзор основных SaaS BI-решений. Таблица 7

|  |  |  |  |
| --- | --- | --- | --- |
| № | Компания | IT-решение | Адрес в Интернете |
| 1 | InetSoft | StyleIntelligence | www.inetsoft.com/products |
| Style Scope |
| Style Report Enterprise |
| 2 | PivotLink | PivotLinkBIPlatform | www.pivotlink.com/betterbi/our-product/bi-platform |
| 3 | Birst | Birst | www.birst.com/products/overview.shtml |
| 4 | Panorama | PanoramaNecto | www.panorama.com/products/business-intelligence-products.html |
| PanoramaNotaView |
| 5 | Mirror 42 | KPILibrary\* | kpilibrary.com |
| KPIBenchmark\* |
| KPIDashboard |
| 6 | KPI Lab | Библиотека показателей\* | www.kpilib.ru |
| \*Решения KPILibrary, KPIBenchmark и Библиотека показателей – это электронные библиотеки ключевых показателей эффективности. Их нельзя причислить к BI-системам в силу того, что они не оперируют данными компании. Однако мы указываем их здесь, так как эти решения напрямую связаны с системами мониторинга ключевых показателей эффективности. | | | |

Отдельно отметим решение *KPILibrary*голландской компании Mirror 42 (№5 в таблице 7), представляющую собой библиотеку ключевых показателей эффективности функционирования организации, которую формируют непосредственно пользователи системы. Они могут добавлять и редактировать показатели, участвовать в голосованиях, касающихся их числовых значений (минимальное, максимальное, оптимальное), устанавливать соответствие показателей определенному типу стратегии (предусмотрено четыре типа стратегии: дифференциация продукта/услуги и лидерство по издержкам, предложенные в работе [[17]](#footnote-18), а также выстраивание доверительных отношений с клиентом и операционное превосходство, предложенные в [[18]](#footnote-19)). В этом решении реализуется инновационный принцип построения информационных систем – web 2.0, заключающийся в том, что система становится тем лучше, чем больше людей ее используют. У *KPILibrary* есть русскоязычный аналог – *Библиотека показателей*.

Кроме растущей потребности в интегрированных ИТ-продуктах, аналитики отмечают рост спроса на «легкие» SaaS-системы, поддерживающие решение отдельных задач системы управления предприятием[[19]](#footnote-20) (например, контроль достижения целей или бизнес-моделирование). Многие из перечисленных систем уже сейчас предоставляются как облачные решения (*SaaS BI-системы, SugarCRM, AuditExpertWeb,BusinessPlanExpert, ComparionSuite, КонсультантПлюс, ComparionTeamTime, ЭСППР ГУ ВШЭ, Мегаплан, GlobalBestPractices, Vistageconnect*). Они хорошо подходят для компаний малого и среднего бизнеса, а также для подразделений крупных компаний, которые не желают тратить значительные средства на внедрение СППР. Использование простых SaaS-решений позволят сохранить гибкость и автономность, а также даст возможность вырабатывать эффективные управленческие решения в постоянно меняющейся среде[[20]](#footnote-21). Кроме того, следует учитывать, что *формирование* стратегии – это не процесс, который должен поддерживаться непрерывно, а проект, имеющий определенные сроки и, в идеале, достаточно редко повторяющийся. Поэтому выгоднее взять ИТ-систему в аренду на время исполнения этого проекта, а не покупать бессрочные лицензии. Данное утверждение не относится к системам, направленным на поддержку *реализации* стратегии, которые следует использовать постоянно, хотя и они (например, системы для мониторинга КПЭ) предоставляются в виде SaaS-решений.

Также распространение получают решения для портативных устройств: планшетных компьютеров и мобильных телефонов. Аналитики компании GartnerGroup предсказывают, что треть функциональных возможностей BI-решений[[21]](#footnote-22) будет в ближайшее время доступна на портативных устройствах. Проблемы, связанные с созданием мобильных систем, интересует не только разработчиков, но и исследователей. Например, визуализации стратегических целей на экране небольшого размера посвящена статья[[22]](#footnote-23).

Эксперты считают, что значительным препятствием в развитии рынка «легких» ИТ-продуктов является слабая осведомленность[[23]](#footnote-24) пользователей об имеющемся предложении. Для решения данной проблемы поставщики предпринимают шаги в сторону активной рекламы своих продуктов (бесплатный пробный период, качественные обучающие материалы, сообщества пользователей), и потенциальные клиенты обращают на это внимание. Кроме того, облачные системы доступны независимо от того, в какой стране находится поставщик, поэтому их выбор шире, чем выбор «коробочных» ИТ-продуктов.

Таким образом, после изучения особенностей модели SaaS, прогнозов её развития и данных о существующих на рынке решениях планируется разрабатывать Web-приложение, предоставляемое компаниям-пользователям (производителям ОАТ) именно на основе этой модели.

### Разработка архитектуры информационной системы SaaS-приложения конкурентного анализа

Разрабатываемое SaaS-приложение имеет клиент-серверный (КС) тип архитектуры информационной системы (ИС). Первая реализация КС-архитектуры (двухуровневая архитектура) представляла собой «толстый» клиент, включающий в себя как пользовательский интерфейс, так и бизнес-логику, который обменивался данными с сервером баз данных. Используя данную архитектуру, мы абстрагируемся от внутреннего представления данных (в отличие от архитектуры файл-сервер) и используем их на уровне логической схемы[[24]](#footnote-25).

Основные особенности данного подхода:

* + работа клиента с данными через запросы к серверному ПО;
  + разделение базовых функций приложения между клиентом и сервером.

Однако такой подход не является достаточно приемлемым для решения данной бизнес-задачи. Дело в том, что при подобном подходе не решается крайне важная проблема: поскольку бизнес-логика остается на стороне клиента, то внесение небольших изменений в работу алгоритма требует обновления ПО. Помимо этого можно выделить следующие минусы:

* + слабая защита данных от взлома;
  + высокая сложность администрирования и настройки рабочих мест пользователей системы;
  + высокие требования к производительности ПК на рабочих местах.

Для решения этих проблем была предложена новая модель проектирования КС-архитектуры, а именно – трехуровневая клиент-серверная архитектура. В отличие от двухуровневой архитектуры – появляется еще один слой – сервер приложений. В нашем случае физически он будет располагаться на том же сервере, что и сервер баз данных, однако на уровне логики данная модель представляет собой следующее:

* + на стороне клиента располагается только пользовательский интерфейс, обеспечивающий взаимодействие и передачу данных на серверную часть;
  + сервер приложений содержит в себе бизнес-логику, обрабатывает запросы от клиента, передает их серверу БД, получает ответ и возвращает полученный результат;
  + сервер баз данных обеспечивает управление данными.

Реализация данного подхода иллюстрируется рисунком 40.

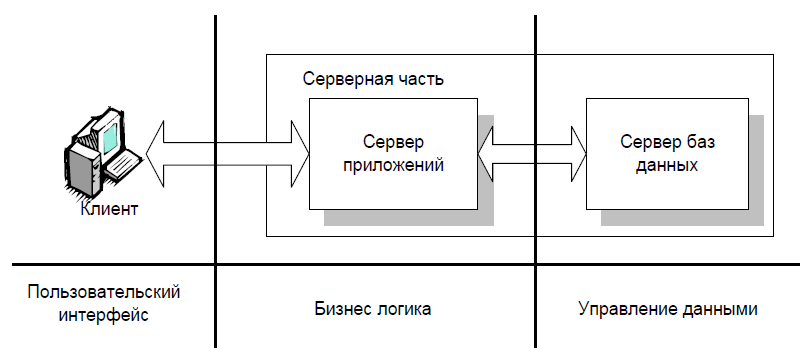


Рис. 40. Трехуровневая клиент-серверная архитектура

Перейдем к рассмотрению плюсов и минусов данной модели. Её плюсами являются:

* при использовании тонкого клиента снижаются требования к ПК клиента; на клиенте расположен только пользовательский интерфейс, доступ к которому пользователь, как правило, получает посредством браузера;
* между клиентской программой и сервером приложений передается лишь минимально необходимый поток данных (аргументы вызываемых функций и возвращаемые ими значения);
* снижение нагрузки на сервер данных и, как следствие, повышение скорости работы системы в целом;
* упрощение наращивания функциональности и обновления ПО.

Минусы:

* Увеличение расходов на администрирование и обслуживание серверной части.

Разрабатываемое SaaS-приложение конкурентного анализа отвечает всем вышеописываемым характеристикам. Высокоуровневая архитектура[[25]](#footnote-26) web-системы представлена на рис. 41. Перейдем к ее рассмотрению.

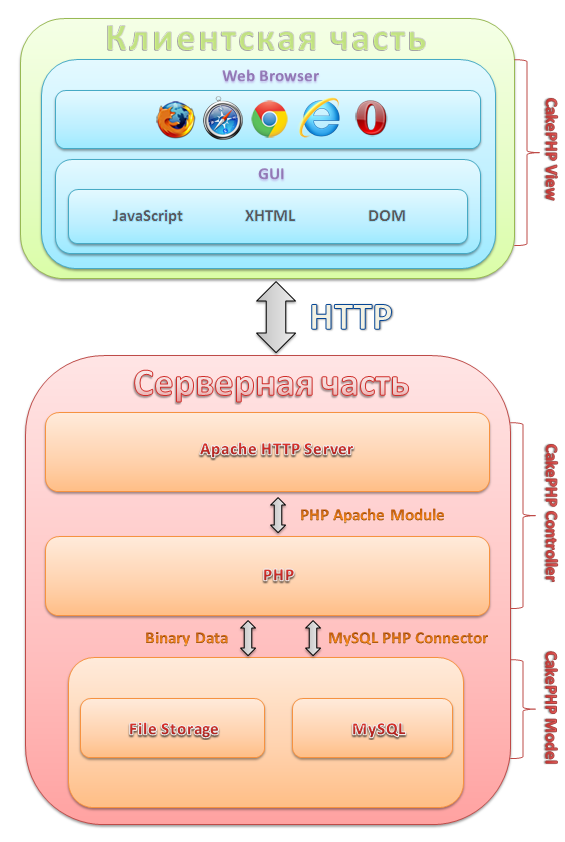


Рис. 41. Высокоуровневая архитектура системы

Доступ к пользовательскому интерфейсу (клиент) будет осуществляться через окно браузера (WebBrowser:MSInternetExplorer, MozillaFirefox, Opera, GoogleChrome, Safari). На наш взгляд, это оптимальное решение, т.к. оно требует от пользователя минимальных затрат, связанных с установкой и обслуживанием ПО. В первое время приложение планируется как бесплатный сервис, однако в дальнейшем возможно введение ежемесячной абонентской платы за использование. Собственно этим и исчерпываются возможные неудобства клиента. Реализация интерфейса будет организована стандартным образом при помощи XHTML-разметки (англ. ExtensibleHypertextMarkupLanguage — расширяемый язык разметки гипертекста), доступ к которой для скриптов (JavaScript) и программ осуществляется посредством DOM (от англ. DocumentObjectModel — «объектная модель документа»).

Передача данных от клиентской части к серверной будет осуществляться при помощи стандартного протокола прикладного уровня передачи данных HTTP (англ. HyperTextTransferPrоtocоl — «протокол передачи гипертекста»).

Серверная часть реализована на основе ApacheHTTPServer, к которому средствами PHPApacheModule подключен скриптовый язык программирования PHP (англ. PHP:HypertextPreprocessor — «PHP: препроцессор гипертекста»). Передача логических данных, обрабатываемых PHP в БД, осуществляется при помощи MySQLPHPConnector. Пользовательский контент передается в файловое хранилище в виде стандартных документов (BinaryData).

В качестве СУБД для сервера БД выбран MySQL. MySQL является решением для малых и средних приложений, что отвечает всем нашим требованиям. Пользовательский контент (файлы с таблицами и диаграммами, изображения, видео и т.д.) будет храниться в файловом хранилище.

Организация бизнес-логики будет отвечать требованиям модели MVC (ModelViewController), которая в свою очередь является проекцией трехуровневой КС-архитектуры. Сам же движок будет запрограммирован средствами PHP (FrameworkCakePHP). Подробнее см. раздел 3.2.3 «Особенности использования фреймворкаCakePHP».

Важно отметить, что как было сказано ранее, мы не будем стремиться к физическому разделению сервера баз данных и сервера приложений, потому как для реализации прототипа SaaS-приложения в рамках демонстрационного прототипа подобный подход является излишним. По тем же причинам мы не станем распараллеливать вычислительные мощности системы. Более подробно о подходе к организации и реализации архитектуры приложения будет сказано в соответствующих главах ниже.

## Разработка интерфейсов проектирования и использования SaaS-приложения «Competition» для конкурентного анализа

### Создание графических моделей SaaS-приложения в виде UML-диаграмм

Первым средством, использованным при проектировании SaaS-приложения, стал инструментарий создания UML-диаграмм. До начала разработки каждой отдельной диаграммы был проведен комплексный анализ унифицированного языка моделирования (UML), рассмотрены для сравнения основные типы диаграмм с целью отбора наиболее актуальных из них. За основу взят стандарт UML 1.5, так как для решения нашей задачи, он является исчерпывающим.

UML 1.5 определяет двенадцать типов диаграмм, разделенных на три группы. Это диаграммы, представляющие:

* статическую структуру приложения
* поведенческие аспекты системы
* физические аспекты функционирования системы

Бабич в своем кратком обзоре UML[[26]](#footnote-27) рекомендует остановиться на рассмотрении всего восьми типов диаграмм, а именно: прецедентов, классов, объектов, последовательностей, взаимодействия, состояний, активности, развертывания. Далее по итогам проведенного сравнительного анализа данный набор был сокращен до четырех диаграмм, в полноте описывающих архитектуру SaaS-приложения. Рассмотрев назначение каждого из ранее выделенных типов диаграмм, мы решили, что для проектирования SaaS-приложения будет вполне достаточно применения следующих диаграмм: *прецедентов, классов, последовательностей* и *активности*. Остальные диаграммы на наш взгляд для решения поставленной задачи не требуются. Теперь перейдем к аргументации нашего выбора относительно каждого типа диаграмм.

Диаграмма *прецедентов* включается в наш набор априори. С ее помощью мы определим границы и контекст моделируемой предметной области, а также сформируем общие требования к поведению проектируемой системы.

Аналогичная ситуация с диаграммой *классов*. С её помощью мы детализируем и конкретизируем составные части будущей системы. Более того, если диаграмма прецедентов отражает начальный этап проектирования, то диаграмма классов завершает проектирование и является отправной точкой процесса разработки.

Диаграмма *последовательностей* является вспомогательной для диаграммы прецедентов. Ее основные функции это детализация логики сценариев использования, а также отображение временных особенностей передачи и приема сообщений объектами.

Диаграмма *активности* представляет собой алгоритм решения поставленной задачи (аналог блок-схемы) и поэтому входит в основной набор используемых диаграмм.

Теперь, что касается оставшихся диаграмм:

Относительно диаграммы *объектов* Бабич пишет следующее: «…диаграмма объектов – это своего рода снимок состояния системы в определенный момент времени, показывающий множество объектов, их состояния и отношения между ними в данный момент». В нашем случае использование подобной диаграммы не целесообразно, потому как поведенческие особенности агентов слишком сложны, и при ее конструировании мы получим либо чрезмерно перегруженную диаграмму, либо напротив, крайне упрощенную диаграмму, описывающую тривиальный случай.

Диаграммы взаимодействий, состояний и развертывания были бы полезны и целесообразны при разработке коммерческого ПО, но т.к. наш проект ограничен демонстрационным прототипом, останавливаться на них не будем.

#### Диаграмма прецедентов

Перейдем к подробному рассмотрению взаимодействия нашей системы с пользователями, в роли которых выступают как сотрудники, поддерживающие разрабатываемую систему, так и внешние пользователи и специалисты. Проанализируем поведение каждого из них, опираясь на диаграммы прецедентов внешней (рис. 42) и внутренней (рис. 43) частей системы.

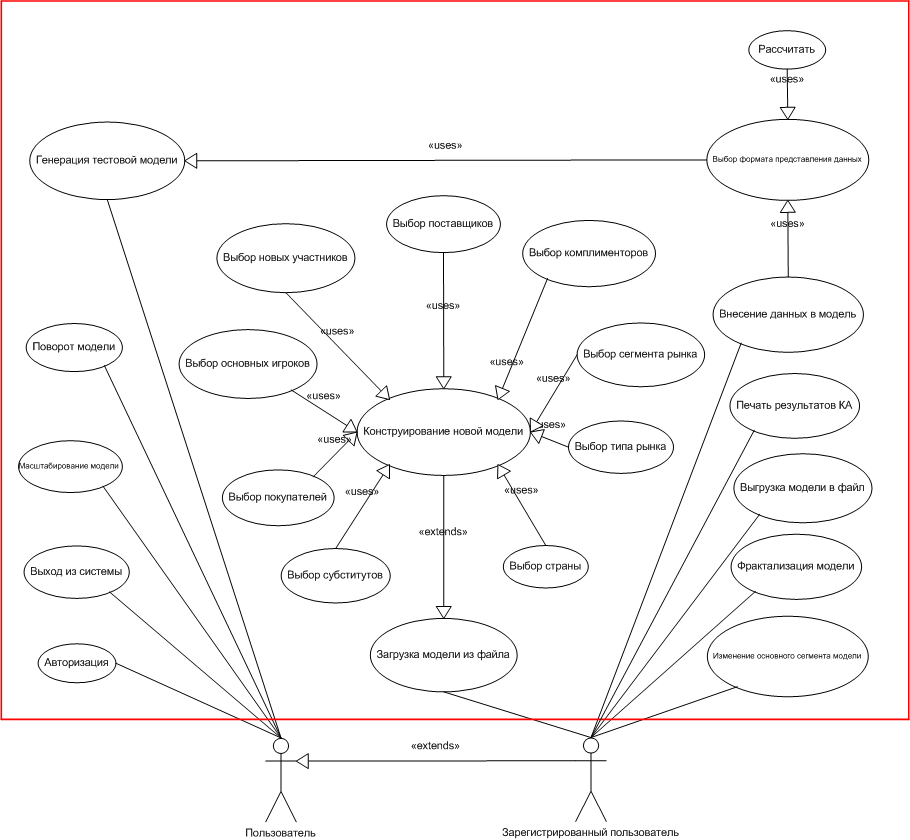


Рис. 42. Диаграмма вариантов использования внешней части системы

1. **Пользователь**.В роли Пользователя, как правило, выступает эксперт в какой-либо области или сегменте рынка. До тех пор, пока он не зарегистрируется в системе и не приобретет статус «Зарегистрированный пользователь», он обладает крайне ограниченным спектром возможностей. По сути, все их возможные действия ограничиваются генерацией тестовой модели и ряда простейших операций над ней. Тестовая модель создается в ходе проектирования демонстрационного прототипа и представляет собой сегмент рынка авиапромышленности, а именно БПЛА. Пользователь может ознакомиться с ней, изменить графическое расположение сил и агентов (без возможности редактировать связи между ними, а также их параметры), повернуть ее, масштабировать. Масштабирование модели является одной из наиболее важных графических составляющих модели, так как в случае расчета крупного сегмента рынка мы можем иметь дело с большой иерархической вложенностью (фрактализацией), большим количеством сил, агентов и связей между ними. В таком случае возможность приблизить и отдалить какой-либо сегмент модели является крайне необходимой. Также Пользователь может пройти процедуру авторизации или выхода из системы.
2. **Зарегистрированный пользователь**. Зарегистрированный пользователь, по сути, является тем же самым Пользователем, описанным выше, однако имеет расширенный спектр возможностей, таких как: генерация собственной модели из исходного набора агентов и параметров, изменение основного сегмента модели, ее фрактализация, возможность редактировать исходные параметры агентов (внесение собственных данных в модель), а также ряд функций по работе с моделью (выгрузка модели в файл, печать результатов конкурентного анализа, сохранение и загрузка модели из БД). При этом роль Зарегистрированного пользователя содержит в себе все возможности Пользователя.

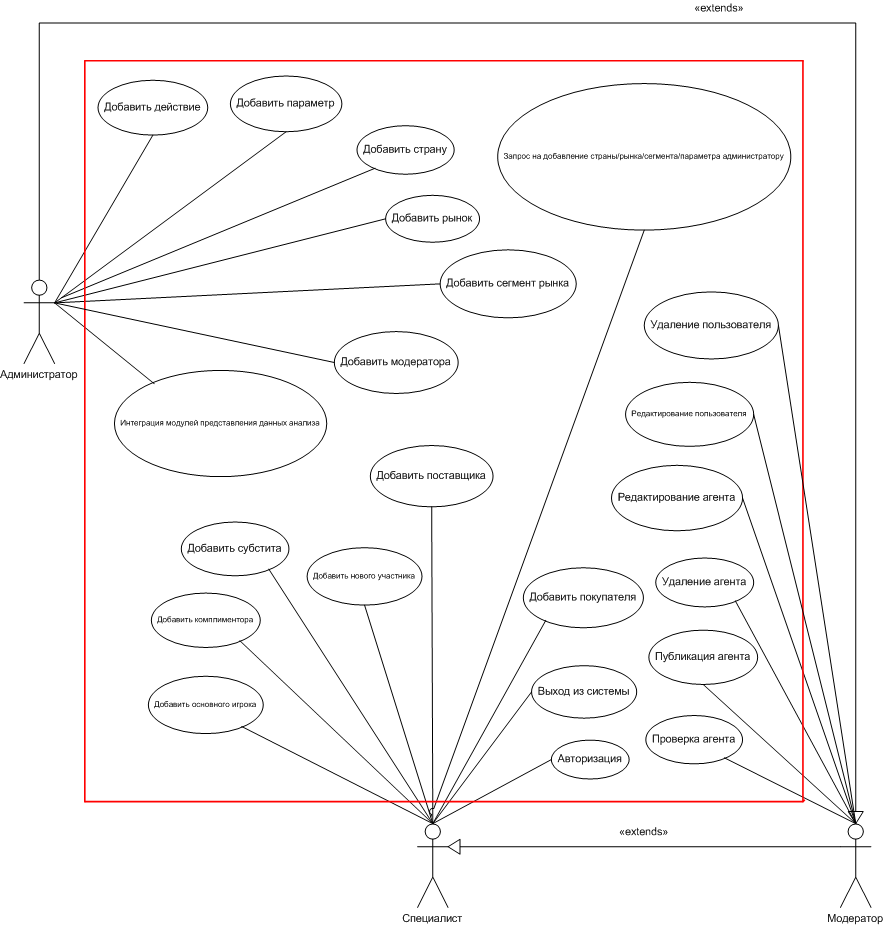


Рис. 43.Диаграмма вариантов использования внутренней части системы

1. **Специалист**. В роли Специалиста выступает эксперт в определенной области (или компания-поставщик/производитель). Он является основным накопителем и поставщиком информации, вносимой в систему, которая в дальнейшем проверяется модератором. Если в роли Специалиста оказывается компания-заказчик моделирования, то его действия связаны с добавлением этой компании, ее основных конкурентов, поставщиков (если они не были внесены ранее) и т.д. в разработанную систему, с одной единственной целью – в дальнейшем самому же воспользоваться указанными данными для конкурентного анализа в интересующем его сегменте рынка. Специалист обладает так же и более широкими возможностями по добавлению страны/рынка или сегмента рынка в том случае, если ранее данная информация внесена не была.
2. **Модератор**. Его действия связаны с проверкой достоверности информации, вносимой специалистом. Если информация оказывается подлинной (Специалистом предоставлены соответствующие документы, которые подтверждают наличие того или иного агента, или же подлинность установлена самим модератором), то она вносится в систему (например, добавляется новый Основной Игрок рынка); в противном случае добавление информации отклоняется, а Специалист получает письмо с причинами отказа о внесении. Так же, Модератор следит за актуальностью информации об агентах и принимает соответствующие решения и действия связанные либо с редактированием агента (если о нем изменилась информация, скажем, он перестал быть Основным Игроком на рынке), либо с его удалением из системы (в случае, если, скажем, какая-то компания обанкротилась).
3. **Администратор**. Основная роль Администратора – добавление (назначение) Модераторов, а также контроль их деятельности. Помимо этого Администратор рассматривает и обрабатывает ряд запросов от Специалиста, например, запрос на добавление неучтенного ранее параметра какому-либо агенту, или добавление ранее отсутствующего в системе сегмента рынка или же исполняет их по собственной инициативе.

Роль Администратора содержит в себе все возможности Модератора и Специалиста. Роль Модератора содержит в себе все возможности Специалиста.

#### Диаграмма классов

Подробнее остановимся на диаграмме классов, т.к. при её проектировании был разработан переход от ССС (гл.1) и математических моделей, описывающих взаимодействие элементов системы (агентов) (гл.2) к взаимодействию объектов различных классов в структуре SaaS-приложения.

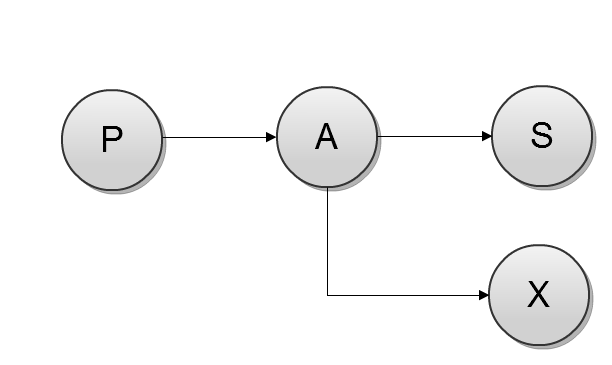


Рис. 44. Основные сущности при переходе от CCС к программному приложению

Взаимодействие и порядок инициации основных сущностей изображен на рис. 44. В первую очередь задается набор элементов системы (силы в расширенной модели М. Портера): где *k* - тип силы (основные конкуренты, поставщики, покупатели и т.д.), *i -* уровень иерархии, и связи между ними (класс power\_powers (PP)). Затем в соответствие к каждому элементу систем (силе) создается агент , где индексы *k* и *i* соответствуют индексам силы, *j* – порядковый номер агента в множестве подобных. Далее для каждого агента задается набор влияний (класс agent\_agents (AA)). Классы PP и AA по сути представляют собой двумерные матрицы (рис. 45).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | - | 1 | 0 | … | 1 |
|  | 1 | - | 0 | … | 0 |
|  | 0 | 0 | - | … | 1 |
|  | … | … | … | - | … |
|  | 1 | 0 | 1 | … | - |

Рис. 45. Матрица взаимных влияний агентов

Набор стратегий задан строго – четырьмя отдельными строками в БД. Действия задаются по параметру name (рис. 46). При выгрузке данных из БД происходит сравнение по данному параметру, и вызывается соответствующая функция.

Все связи задаются через вспомогательные таблицы. Каждому агенту с их помощью ставится в соответствие набор действий, а так же стратегий (рис. 46).

Названия классов соответствует нотации CakePHP (см. модели CakePHP в разделе «Особенности использования фреймворка CakePHP).

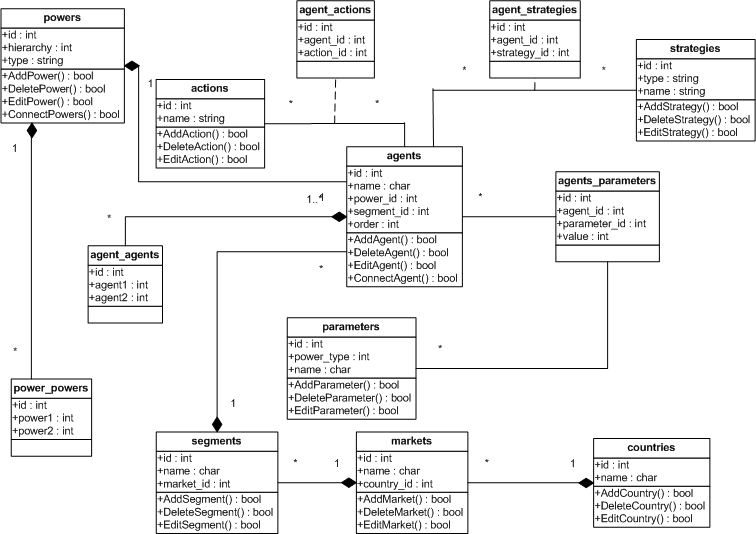


Рис. 46. Представление диаграммы классов

Перейдем к подробному описанию элементов данной диаграммы:

1. **Класс powers.** Содержит атрибуты hierarchy – степень фрактализации и type – тип силы и добавляемых агентов (напр., «Новые игроки», «Поставщики» и т.д.), а также стандартные операторы, доступные эктору «Пользователь» – добавить\_силу(), удалить\_силу(), отредактировать\_силу(), связать\_силы(). Одной силе соответствует несколько агентов.
2. **Класс powers\_powers.** Вспомогательный класс для powers. Содержит атрибуты power1 и power2, обозначающие связь между силой1 и силой2.
3. **Класс agents.** Основной класс проектируемого SaaS-приложения. Так или иначе, с ним связаны все остальные классы системы. Содержит атрибуты name – имя объекта, order – порядковый номер агента в силе, а также стандартные операторы, доступные пользователю «Модератор» – добавить\_агента(), удалить\_агента (), отредактировать\_агента() и «Пользователь» – связать\_агентов(). Одному агенту соответствует несколько действий, параметров и связей, а также единственная стратегия и единственный сегмент рынка.
4. **Класс actions.** Содержит атрибут name – название действия, по которому вызывается соответствующая функция, участвующая в получении результата конкурентного анализа сегмента рынка, а также стандартные операторы, доступные эктору «Администратор» – добавить\_действие (), удалить\_действие(), отредактировать\_действие().
5. **Класс agent\_actions.** Вспомогательный класс для agents и actions. Содержит атрибуты agent\_id и action\_id, обозначающие привязку нескольких возможных действий к одному агенту.
6. **Класс agent\_agents.** Вспомогательный класс для agents. Содержит атрибуты agent1 agent2, обозначающие связь между агентом1 и агентом2.
7. **Класс strategies.** Содержит атрибуты type – тип стратегии и name – имя стратегии (например, стратегия «Лидер»), а также стандартные операторы, доступные эктору «Администратор», а именно: добавить\_стратегию(), удалить\_стратегию(), отредактировать\_стратегию(). Выбранная агентом стратегия, определяет его поведение (действия) на рынке. В каждый определенный момент времени одному агенту может соответствовать только одна стратегия.
8. **Класс agent\_strategies.** Вспомогательный класс для agents и strategies. Содержит атрибуты agent\_id и strategy\_id, обозначающие привязку стратегии к агенту.
9. **Класс parameters.** Содержит атрибуты power\_type – тип силы (поскольку тип агента зависит от типа силы, в которую он помещен) и name – название параметра, а также стандартные операторы, доступные пользователю «Администратор» – добавить\_параметр(), удалить\_параметр(), отредактировать\_параметр().
10. **Класс agent\_parameters.** Вспомогательный класс для agents и parameters. Содержит атрибуты agent\_id и parameter\_id, обозначающие привязку параметра к агенту, а также значение параметра – value.
11. **Класс segments.** Класс, помогающий осуществлять фильтрацию агентов по определенному сегменту рынка (в нашем случае это БПЛА). Содержит атрибут name – название сегмента рынка, а также операторы, доступные эктору «Администратор», а именно: добавить\_сегмент(), удалить\_сегмент(), отредактировать\_сегмент().Связан с классом markets. Одному сегменту соответствует несколько агентов.
12. **Класс markets.** Класс, помогающий осуществлять фильтрацию агентов по определенному типу рынка (в нашем случае это Авиастроение). Одному рынку может соответствовать несколько сегментов рынка. Содержит атрибут name – название типа рынка, а также операторы, доступные эктору «Администратор», а именно: добавить\_рынок(), удалить\_рынок(), отредактировать\_рынок ().Связан с классом countries.
13. **Класс countries.** Класс, помогающий осуществлять фильтрацию агентов по стране (в нашем случае это Российская Федерация). Одной стране может соответствовать несколько типов рынка. Содержит атрибут name – название типа рынка, а также операторы, доступные эктору «Администратор», а именно: добавить\_страну(), удалить\_страну(), отредактировать\_страну ().

#### Диаграммы последовательностей

Перейдем к рассмотрению последовательности добавления нового агента в проектируемую систему. Для этого мы продемонстрируем процесс взаимодействия и линии жизни четырех объектов, показанных на рис. 47, на примере совершаемых ими действий.

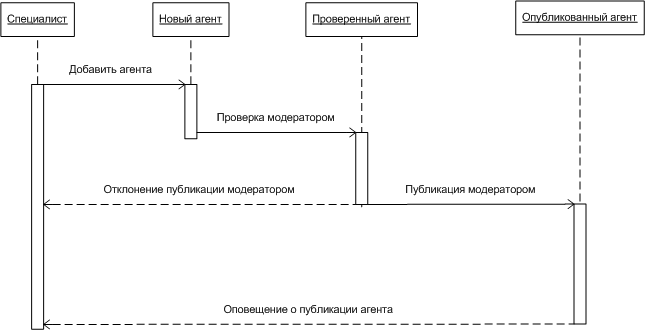


Рис. 47. Диаграмма последовательностей (добавление агента в систему)

1. **Специалист.** Деятельность Специалиста не прерывается на протяжении всего времени выполнения действий прочими объектами. Именно он совершает первое действие в последовательности – добавляет агента в систему (информацию о новом агенте), после чего ожидает результата проверки достоверности объекта, тем самым порождая объект «Новый агент».
2. **Новый агент.** Данный объект может соответствовать одному из шести типов сил модифицированной модели М. Портера, а именно: *основные игроки, покупатели, поставщики, новые игроки, субституты* или *комплементоры.* Этот объект содержит в себе информацию о реальном игроке того или иного сегмента рынка (принадлежность к сегменту, тип, имя, набор параметров, официальные документы, подтверждающие его достоверность). Существование Нового агента состоит из трех этапов:

* формирование и добавление информации об агенте в систему;
* получение и обработка информации модератором;
* вынесение решения о публикации или отклонении публикации агента модератором.

Если принимается решение об отклонении публикации, то об этом оповещается Специалист, и процесс завершается. В противном случае, если принимается решение о публикации агента в систему (установлена его достоверность), создается новый объект «Проверенный агент», полностью соответствующий Новому агенту, но с дополнительной пометкой об успешно пройденной проверке.

1. **Проверенный агент.** Время существования Проверенного агента не велико и соответствует тому времени, которое требуется для внесения соответствующей информации в базу данных и сохранении приложений (сканы официальных документов, подтверждающих достоверность агента) в файловое хранилище. После публикации этого объекта модератором, появляется новый объект – «Опубликованный агент».
2. **Опубликованный агент.** Появление данного объекта говорит об успешном прохождении последовательности действий, в ходе которых в систему был добавлен и опубликован новый агент.

### Проектирование архитектуры БД

В данном разделе мы подробно рассмотрим построение базы данных (БД) web-системы. Схема БД представлена на рис. 48.

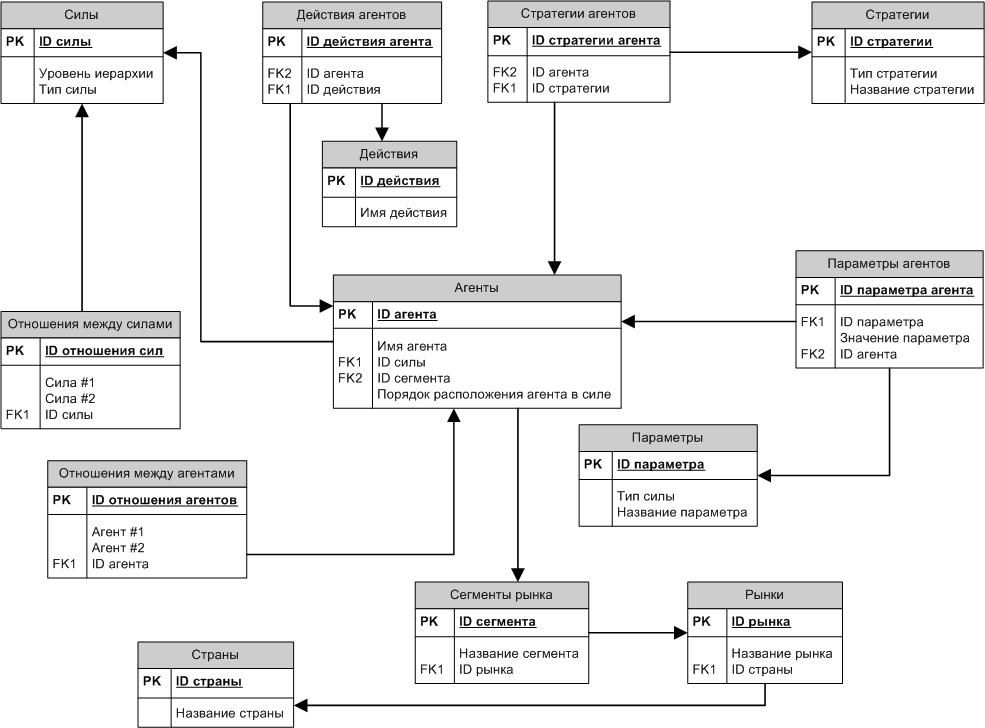


Рис. 48. Схема базы данных системы

Выделим в схеме БД основные и вспомогательные таблицы и дадим пояснение каждой из них. Наиважнейшей сущностью является таблица «Агенты». Именно вокруг нее и строится вся система, а на ее основе производится конкурентный анализ сегмента рынка.

1. **Силы.** Одна из основных таблиц. Является «каркасом» для данных из таблицы «Агенты». По сути, при расчете модифицированной модели М. Портера несет на себе функции определения дополнительных характеристик агента, его позиционирование (уровень иерархии (фрактализация) и тип агента). К одной силе может быть привязано несколько агентов.
   1. **Отношения между силами.** Данная таблица, по сути, является способом хранения разряженной матрицы отношений между различными силами. При конкурентном анализе, нам важно учитывать все связи между силами; именно эта информация и хранится в этой таблице.
2. **Агенты.** Основная таблица нашей web-системы. В ней хранится список агентов с указанием имени, позиции, уровня фрактализации (через таблицу Силы) и принадлежности к определенному сегменту рынка (для фильтрации агентов). Она является центральной, и с ней так или иначе связаны все прочие таблицы. Конкурентный анализ модифицированной модели М. Портера происходит на основе данных параметров агентов, выбранных ими стратегий (а следовательно, и действий) и отношений между ними.
   1. **Отношения между агентами.** Данная таблица, по сути, является способом хранения разряженной матрицы отношений между различными агентами. При конкурентном анализе, нам важно учитывать все связи между агентами; именно эта информация и хранится в этой таблице.
   2. **Параметры агентов.** Данная таблица связывает конкретного агента с набором его параметров и их значениями.
      1. **Параметры.** Это одна из наиболее важных таблиц, поскольку набор параметров того или иного агента играет основополагающую роль в ходе конкурентного анализа сегмента рынка. Для каждой силы определен свой набор параметров, одинаковый для каждого агента, привязанного к силе (различаются только значения). Например, для покупателей определены такие параметры, как *спрос, бюджет* и *полезность.*
   3. **Действия агентов.** Данная таблица связывает конкретного агента с набором его возможных действий.
      1. **Действия.** Еще одна крайне важная таблица. В ней содержатся имена различных действий, предусмотренных для агентов. Каждое действия, на основе его имени, соответствует той или иной функции, запрограммированной в вычислительном модуле на языке PHP. Таким образом, каждое действие по своей сути является вызовом определенной функции, работающей с параметрами агентов.
   4. **Стратегии агентов.** Данная таблица связывает конкретного агента с выбранной им стратегией.
      1. **Стратегии.** В данной таблице содержится набор из трех основных стратегий – «лидер», «следующий за лидером», «бросающий вызов лидеру», определяющих поведение, соответствующее таблице «Действия», того или иного агента.
3. **Сегменты рынка.** Вспомогательная таблица для фильтрации агентов. Определяет профиль агента (в нашем случае это БПЛА). Поскольку наша система будет содержать большое количество различных агентов из самых разных областей и сегментов рынка, то крайне важно точно знать, какой агент к какому сегменту принадлежит.
   * 1. **Рынки.** Более высокий уровень фильтрации, также вспомогательная таблица. Определяет тип рынка, к которому относится агент (в нашем случае это Авиастроение).
        1. **Страны.** Более высокий уровень фильтрации, также вспомогательная таблица. Определяет страну, к которой относится агент (в нашем случае это Российская Федерация).

### Структурный анализ системы на основе диаграмм потоков данных (DFD)

В данном разделе мы воспользуемся методологией графического структурного анализа для описания внешних по отношению к системе источников и адресатов данных, логических функций, потоков данных и хранилищ данных, к которым осуществляется доступ. Построение диаграмм потоков данных существенно поможет в проектировании SaaS-приложения, а также наглядно проиллюстрирует внешнее (рис. 49) и внутреннее (рис. 50) устройство информационной системы. Начнем с рассмотрения диаграммы внешнего устройства потоков данных web-системы.

Внешней сущностью является Пользователь. Он взаимодействует с процессами Авторизации и Конкурентного анализа расширенной модели М. Портера. При Авторизации Пользователь делится своей личной информацией и в случае, если эта информация уже содержится внутри БД системы – выполняется вход, если же информация оказывается новой, в таком случае Пользователь регистрируется в системе, а информация о нем сохраняется в накопитель данных – БД системы. При Конкурентном анализе модели М. Портера пользователь передает процессу все данные и настройки сконструированной модели, а именно: *список агентов* и *отношения между ними*, *список сил* и *отношения между ними*, *формат представления результата* и *параметры сохранения модели в БД.* Параллельно с этими данными, процесс запрашивает следующие данные из накопителя данных – БД системы, а именно *стратегии агентов, параметры агентов* и *действия агентов*. На основе всех этих данных происходит их преобразование в результирующие данные, которые подаются двум следующим процессам: Сохранению модели в БД (в случае, если были заданы соответствующие параметры сохранения; при сохранении в БД, запоминаются также и входные данные) и Представлению результатов Конкурентного анализа на основе данных о формате представления результата.

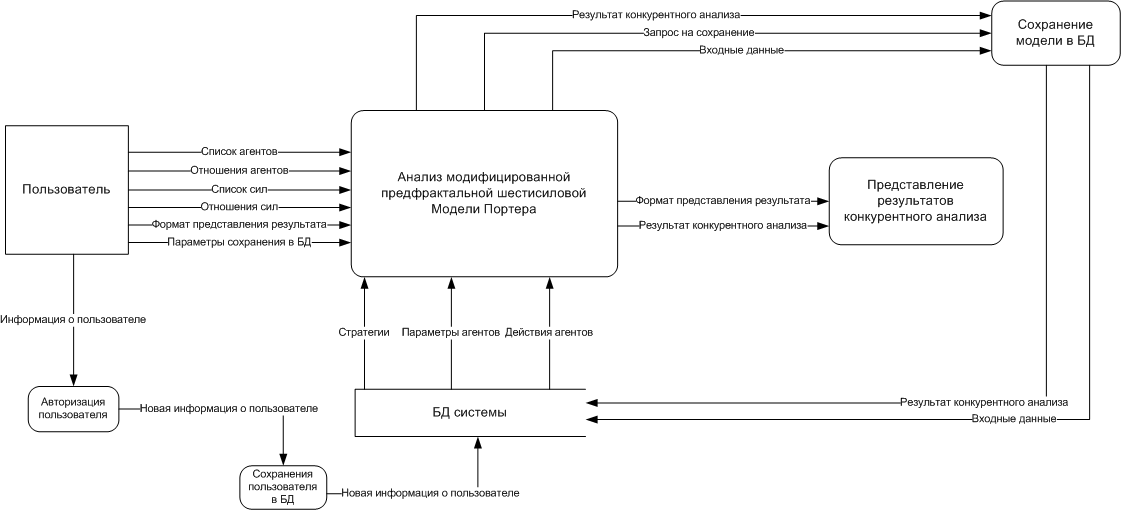
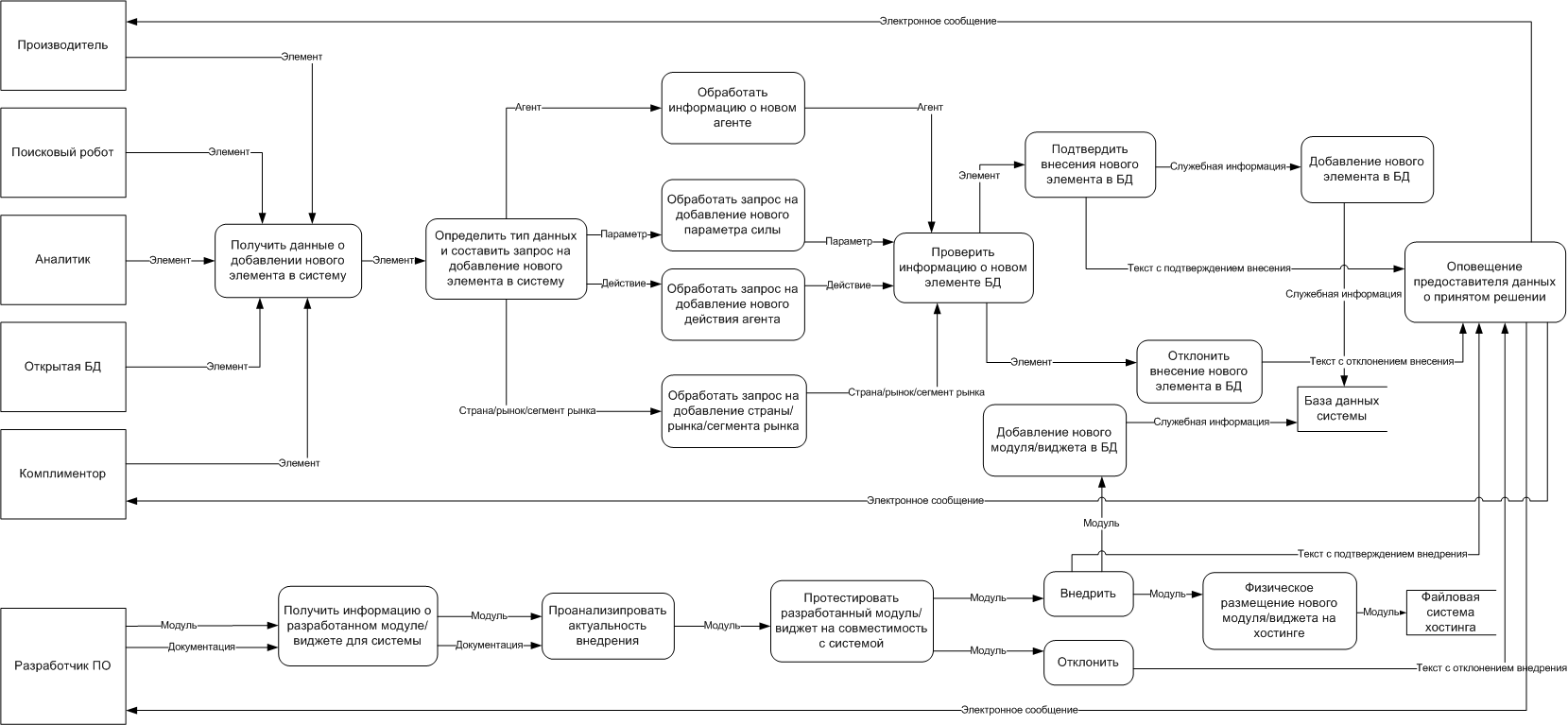
****

Рис. 49. DFD-диаграмма: внешнее устройство потоков данных web-системы

Перейдем теперь к рассмотрению внутреннего устройства потоков данных web-системы. Мы попытались максимально детализировать эту диаграмму, чтобы отобразить всех заинтересованных лиц, использующих систему, а также потенциальные источники данных. Выделены следующие внешние сущности: Производитель, Поисковый робот, Аналитик, Внешняя БД, Комплементор, Разработчик ПО (обозначим их как «Специалист», по аналогии с диаграммой прецедентов). Разработчик ПО поставляет следующие данные в систему: разработанный для интеграции модуль или плагин с подготовленной документацией и описанием. Эти данные проходят через последовательность процессов, связанных с их получением, анализом актуальности использования, тестированием на предмет совместимости с разработанной web-системой и решением о внедрении. Данные о результате решения о внедрении передаются Разработчику по электронной почте. В случае положительного решения сам модуль попадает в накопитель данных – файловую систему хостинга , а служебная информация (место его физического размещения и пр.) в накопитель данных БД системы.

Специалист подает в систему данные, обозначенные как «элемент», тип которых определяется процессом определения типа поступивших данных и составления запроса на их добавление. Элемент может быть *новым агентом, параметром, действием, сегментом рынка, рынком* или *страной.* Как только процесс определения типа данных обработал их и составил запрос на их добавление, определенные данные входят в процесс проверки их достоверности, в результате которого специалист получает данные о принятом решении по электронной почте (за исключением Поискового робота, Открытой БД и Аналитика, поскольку первые две сущности не нуждаются в оповещении, а Аналитик является работником нашей системы) а данные о новом элементе системы переходят в процесс добавления нового элемента в БД и вносятся в накопитель данных – БД системы.

****Рис. 50. DFD-диаграмма: внутреннее устройство потоков данных web-системы

### Описание интерфейсов пользователя (ввод данных и вывод результатов)

Важным этапом проектирования и дальнейшей реализации web-системы является тщательно продуманная информационная архитектура, а также подготовка и составление макетов пользовательского интерфейса (GUI). На рис. 49 представлен основной макет интерфейса – главная страница с рабочей областью.

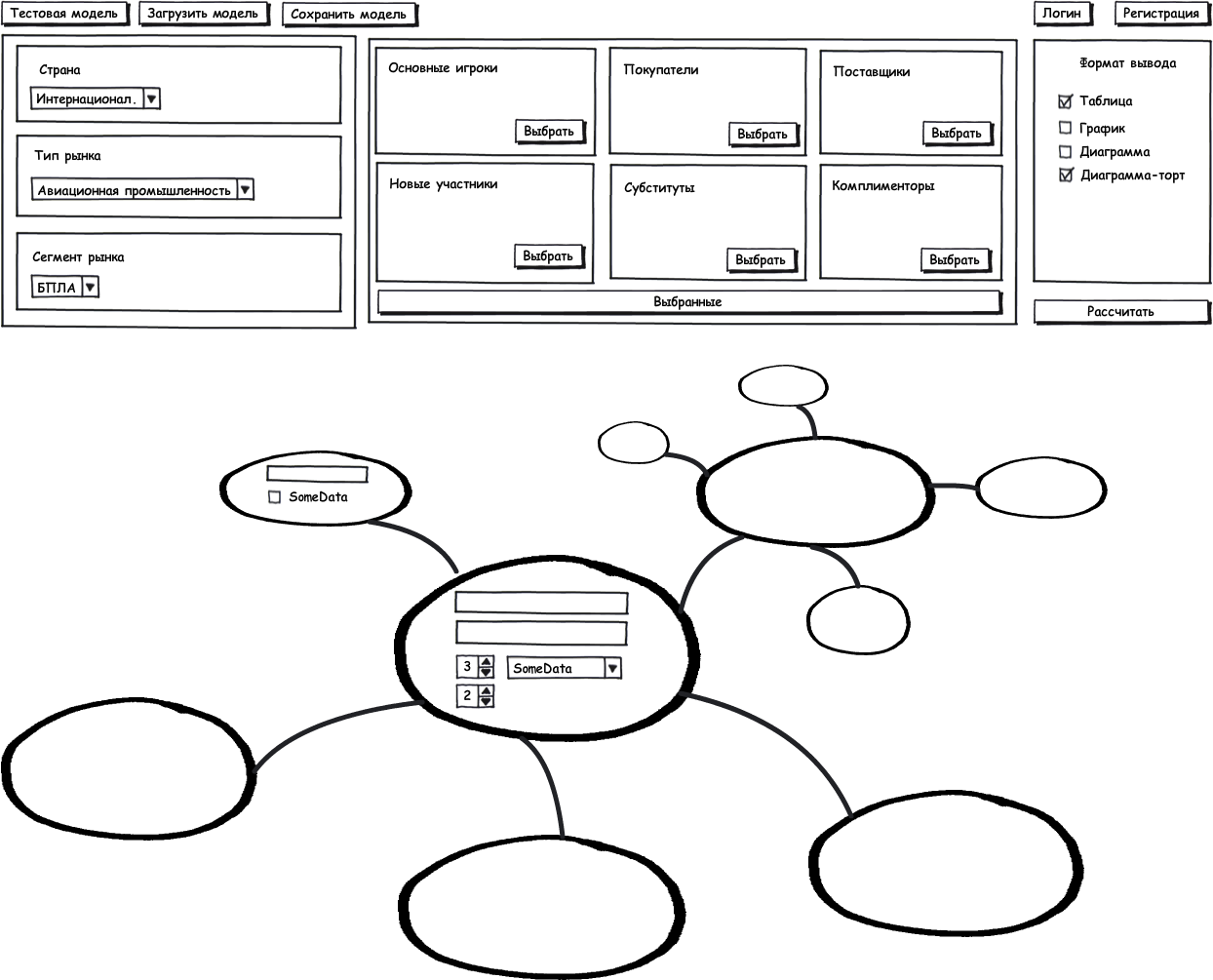
****

Рис. 49. Основной макет интерфейса: главная страница с рабочей областью

Рассмотрим макет подробнее, остановившись на каждом из спроектированных блоков. Верхнее меню для удобства разбито на две части:

* первая отвечает за работу с модифицированной моделью М. Портера («Тестовая модель», «Загрузить модель», «Сохранить модель»);
* вторая же отвечает за взаимодействием с пользователем («Логин», «Регистрация»; после входа в систему меняются на «Личный кабинет» и «Выход»).

Большой блок настроек и фильтров, следующий сразу за основным меню, разбит на три части:

* В первой части происходит фильтрация агентов по местоположению, типу рынка и сегменту рынка. В случае выбора страны «Интернационал» (что соответствует международному конкурентному рынку) предлагается возможность выбора как всех стран, так и только некоторых из них. Данный блок позволяет пользователю сузить круг возможных агентов до отдельного сегмента рынка. Такая возможность является необходимой, потому как наша web-система должна будет располагать очень большим количеством возможных для выбора агентов и возможностями детализации и масштабирования.
* Во второй части происходит отбор агентов для проведения конкурентного анализа. При добавлении того или иного агента к силе, происходит автоматическая фильтрация по типу силы (например, если мы работаем с силой «Новые участники», то среди агентов также будут только «Новые участники»). К каждой силе можно привязать несколько разных агентов.
* В третьей части выбирается формат представления данных, полученных в результате конкурентного анализа.

Далее разворачивается сама модифицированная модель М. Портера, соответствующая выбранным силам и привязанным к ним агентам. Следующим этапом является связь между силами и агентами, после чего возможно проведение конкурентного анализа в заданном сегменте рынка и вывод результатов в выбранном формате.

На данном макете блок настроек и фильтров представлен в развернутом виде, однако при реализации дизайна средствами HTML, CSS и JavaScript он будет свернут в один единый блок, навигация по которому будет производиться с помощью перехода между вкладками, вынесенными в подменю.

На рис. 50 показано вспомогательное всплывающее окно, разворачивающееся по нажатию на кнопку «Выбрать».

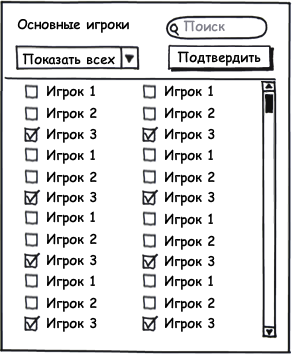


Рис. 50. Окно выбора агентов

В конкретном случае речь идет о вкладке «Основные игроки». Нашему рассмотрению представлен полный список позиций (в данном случае – игроков), предварительно внесенных в базу данных системы, с возможностью фильтрации по поисковой строке на основе полнотекстового поиска с использованием ajax-технологий. Также, помимо отображения всех позиций, предусмотрено отображение только отмеченных или, напротив еще *не* отмеченных позиций. Кнопка «Подтвердить» сохраняет выбранные позиции для последующего проведения конкурентного анализа.

## Применение SaaS-приложения «Competition» при проектировании конкурентной стратегии производителя ОАТ

### Методика и программные средства измерения параметров сектора авиационной промышленности с помощью СПО Avalanche

При проектировании конкурентной стратегии производителя ОАТ, в частности БПЛА, важную роль играет сбор адекватных данных о конкурентах и конкурентной среде. Поэтому данный параграф решено было посвятить описанию методики сбора и обработки данных по сектору АП с помощью системы интернет-мониторинга и конкурентной разведки Avalanche.

Данное семейство информационно-поисковых программ[[27]](#footnote-28) можно выделить в ряду программных средств, используемых аналитиками для Web-мониторинга. В отличие от большинства разработчиков аналитических программ, создатели Avalanche сосредоточились не на детальной обработке полученных документов, а на их поиске и первичной рубрикации.

Фактически, технология Avalanche базируется на трех "китах":

* концепции "умных папок" (Smart Folders);
* автономном интеллектуальном поисковом роботе ("пауке");
* встроенной базе данных, допускающей преобразование в "персональную энциклопедию".

"Умные папки" – это рубрикатор с расширенными возможностями, не только отображающий документы, найденные в Интернете, но и хранящий дополнительную информацию, конкретизирующую смысловой шаблон (по сути, структура папок отражает модель предметной области в том виде, в котором она существует в голове аналитика). Работа с "умными папками" напоминает обычную работу с каталогами в проводнике Microsoft Windows. Однако "умные папки" позволяют сортировать найденные документы по словам и фразам, по стоп-словам, по датам, по выражению и по подобию.

Поисковый робот предназначен для поиска в Сети информации по заранее заданному расписанию, прохода по нескольким уровням ссылок и оценки адекватности найденных ссылок, загрузки найденной информации на локальный компьютер пользователя. Изюминкой робота Avalanchе является возможность автоматически расширять зону поиска для последующих запросов (в любой другой системе набор информационных источников определен и фиксирован заранее). Роботы Avalanche относятся к классу т.н. «проникающих» роботов, способных передвигаться по Интернету не только по гипер-ссылкам (как в большинстве поисковых систем), но также «по бездорожью» (невидимому или «глубинному» Интернету) - сканированием адресного пространства порталов, перебором компонентов индексов баз данных, распознаванием расширяемых адресных наборов и др. Как результат, на крупных порталах робот Avalanche может находить большее количество страниц, нежели любая поисковая система, включая Google.

С помощью Avalanche можно провести экспорт имеющейся базы данных в персональную HTML-энциклопедию, что позволяет в удобном виде представить и хранить информацию в форме персонального Интернет-сайта.

Производитель ОАТ с помощью пакета Avalanche сможет решать основные задачи для своего бизнеса:

* вести мониторинг своей популярности, автоматически собирая все свежие упоминания о фирме в Интернет;
* автоматически пополнять досье на основных конкурентов, мгновенно фиксируя появление любых новых материалов;
* фильтрует результаты поиска других поисковых систем, устраняя ненужные ссылки (например, упоминания о своей фирме в отзывах, статьях и т.д. ) - такая настройка тоже есть в Аваланче.

Ниже представлен разработанный макет использования персональной версии программы Avalanche для мониторинга ситуации в секторе поставок БПЛА. Предполагается, что действия аналитика будут представлены в виде последовательности трех основных шагов:

1. Аналитик определяет границы предметной области в виде набора вложенных папок, описывающих основные объекты анализа: «Беспилотные летательные аппараты», «Истребители 5-го поколения», «Микродирижабли» и т.д. (итоговая настройка будет затем представлена в окне «Дерево рубрик»).

2. Далее аналитик определяет основные сайты в Интернете, с которых роботу следует начинать поиск новых документов по заданной тематике, например http://bp-la.ru (Этот список виден на рисунке в окне «Список сайтов»). Впоследствии робот будет самостоятельно расширять зону поиска, находя и собирая новые интересные ссылки.

3. Далее по заданному регламенту (например, каждое утро в 8:00) робот Avalanche будет отправляться в Интернет, обходя заданный список источников и собирая новые документы. Потом начинают работать умные папки, и каждая разбирает и сохраняет себе относящиеся к ней документы. И когда к экрану подойдет аналитик, он увидит окно результатов, стилизованное под окно типичного почтового клиента, где свежие документы будут отображаться уже разложенными по рубрикам.

Конечно, стопроцентная полнота и релевантность результатов поиска в Интернете в принципе недостижима. Но такой легкий и гибкий инструмент, как Avalanche избавляет аналитика от рутины, позволяет собрать большое количество данных за небольшой промежуток времени, а результаты Интернет-поиска делает более точными и удобными для работы.

### Сценарии использования SaaS-приложения «Competition»

### Примеры использования SaaS-приложения «Competition» для проектирования стратегии производителя объектов боевой авиации

# Выводы по главе 3

1. http://book.cakephp.org/1.3/ru/view/890/%D0%A7%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-Model-View-Controller [↑](#footnote-ref-2)
2. Ajax в действии Ajax in Action Автор: Дейв Крейн, Эрик Паскарелло, Даррен Джеймс Издательство: Вильямс ISBN 978-5-8459-1034-9, 1-9323-9461-3; 2008 г. [↑](#footnote-ref-3)
3. http://book.cakephp.org/1.3/ru/view/890/%D0%A7%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-Model-View-Controller [↑](#footnote-ref-4)
4. ГОСТ Р ИСО МЭК ТО 10032-2007: Эталонная модель управления данными (идентичен ISO/IEC TR 10032:2003 Information technology — Reference model of data management) [↑](#footnote-ref-5)
5. Дейт К. Дж. Введение в системы баз данных = Introduction to Database Systems. — 8-е изд. — М.: Вильямс, 2005. — 1328 с. — ISBN 5-8459-0788-8 (рус.) 0-321-19784-4 (англ.) [↑](#footnote-ref-6)
6. http://ru.wikipedia.org/wiki/CakePHP [↑](#footnote-ref-7)
7. http://ru.wikipedia.org/wiki/CamelCase [↑](#footnote-ref-8)
8. http://book.cakephp.org/1.3/ru/view/1522/Code-Generation-with-Bake [↑](#footnote-ref-9)
9. http://php.net/manual/ru/function.mysql-query.php [↑](#footnote-ref-10)
10. http://book.cakephp.org/1.3/ru/view/1017/%D0%9F%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8 [↑](#footnote-ref-11)
11. http://book.cakephp.org/1.3/ru/view/1031/%D0%A1%D0%BE%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8 [↑](#footnote-ref-12)
12. http://book.cakephp.org/1.3/ru/view/1035/Deleting-Data [↑](#footnote-ref-13)
13. Зенкин Д. Приложение в аренду. Статья из приложения к газете «Коммерсантъ» №51 (4106) от 24.03.2009. http://www.kommersant.ru/doc-y.aspx?DocsID=1138954. [↑](#footnote-ref-14)
14. Шашенкова Е. Будущее у SaaS светлое. Статья на портале MSKIT.ru. Дата публикации: 19.02.2010. http://www.mskit.ru/news/n70385/ [↑](#footnote-ref-15)
15. В скобках указаны названия предлагаемых ИТ-решений. [↑](#footnote-ref-16)
16. Business intelligence или сокращенно BI — бизнес-анализ, бизнес-аналитика. Под этим понятием чаще всего подразумевают программное обеспечение, созданное для помощи менеджеру в анализе информации о своей компании и её окружении. [↑](#footnote-ref-17)
17. М. Портер. Конкурентное преимущество. Как достичь высокого результата и обеспечить его устойчивость. – М.: Альпина Бизнес Букс, 2005. – 654 с. [↑](#footnote-ref-18)
18. TreacyM., WiersemaF. TheDisciplineofMarketLeaders.Basic Books, 1997.– 224 p. [↑](#footnote-ref-19)
19. KellyJ.Mega-vendorsdominateGartner's 2010 BI, datawarehouseMagicQuadrants. Tech Target, 4 Feb 2010. АдресвИнтернете: searchbusinessanalytics.techtarget.com/news/1506976/Mega-vendors-dominate-Gartners-2010-BI-data-warehouse-Magic-Quadrants [↑](#footnote-ref-20)
20. LachlanJ.Top 10 BusinessIntelligencepredictionsfor 2011.Dashboard Insight 18 Jan 2011. Адрес в Интернете: www.dashboardinsight.com/articles/new-concepts-in-business-intelligence/top-10-business-intelligence-predictions-for-2011.aspx [↑](#footnote-ref-21)
21. FinleyK. 10 BusinessIntelligenceToolsfortheiPad. 09.02.2011. www.readwriteweb.com/enterprise/2011/02/10-ipad-business-intelligence-tools.php [↑](#footnote-ref-22)
22. CostaC.J., AparicioM.VisualizationofbalancedscorecardonPDAs.SIGDOC 2005. – p. 103–107 [↑](#footnote-ref-23)
23. WiseL.ExpansionOfSaaS-BasedBISolutions. Dashboard Insight, 30 Sep 2009. Адрес в Интернете: www.dashboardinsight.com/articles/new-concepts-in-business-intelligence/expansion-of-saas-based-bi-solutions.aspx [↑](#footnote-ref-24)
24. Лекции по архитектуре информационной системы. <http://it-claim.ru/Education/Course/ISDevelopment/Lecture_3.pdf> [↑](#footnote-ref-25)
25. Розенфельд Л., Морвиль П. Информационная архитектура в Интернете, 2-е издание. – СПб: Символ-Плюс, 2005. – 544с. [↑](#footnote-ref-26)
26. Бабич А. В. UML: Первое знакомство. Пособие для подготовки к сдаче теста UM0-100 (OMG Certified UML Professional Fundamental). – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2011. – 176с. [↑](#footnote-ref-27)
27. Доронин А.И. Бизнес-разведка. – 2-е изд., перераб. И доп. – М. Издательство «Ось-89», 2003. – 384 с. [↑](#footnote-ref-28)