МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Кемеровский государственный университет»

Институт фундаментальных наук

Кафедра ЮНЕСКО по информационным вычислительным технологиям

Диденко Андрей Александрович

Плагин «Принятие решений» в системе управления проектами «JIRA»

Выпускная квалификационная работа

(бакалаврская работа)

по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

направленность (профиль) подготовки «Информационные системы и базы данных»

|  |  |
| --- | --- |
|  |  |
|  | Научный руководитель: |
|  | к.ф.-м.н., доцент, К.С. Иванов |
|  | Работа защищена с оценкой: |
|  |  |

Протокол ГЭК №\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

от «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г.

Секретарь ГЭК\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Кемерово 2017

# Оглавление

[Оглавление 2](#_Toc485340762)

[Введение 4](#_Toc485340763)

[1. Обзор литературы и подготовка инструментария 6](#_Toc485340764)

[1.1. Обзор существующих решений 6](#_Toc485340765)

[1.1.1. «Todoist» 6](#_Toc485340766)

[1.1.2. «Any.do» 8](#_Toc485340767)

[1.1.3. «GanttPro» 9](#_Toc485340768)

[1.1.4. «Asana» 11](#_Toc485340769)

[1.1.5. «Easy Redmine» 13](#_Toc485340770)

[1.1.6. «Freedcamp» 14](#_Toc485340771)

[1.1.7. «Bitrix24» 15](#_Toc485340772)

[1.1.8. «Podio» 17](#_Toc485340773)

[1.1.9. «Wrike» 18](#_Toc485340774)

[1.1.10. «Jira» 20](#_Toc485340775)

[1.2. Обзор способов отображения информации по проекту 22](#_Toc485340776)

[1.2.1. Диаграмма «Гантта» 22](#_Toc485340777)

[1.2.2. Ментальная карта 23](#_Toc485340778)

[1.2.3. Деревья 24](#_Toc485340779)

[1.3. Обзор литературы по теме и подготовка инструментов 25](#_Toc485340780)

[1.3.1. Сущности project, issue и связи между ними 25](#_Toc485340781)

[1.3.2. «Atlassian JIRA»: обзор, реализация, лицензия, архитектура и решаемые проблемы 27](#_Toc485340782)

[1.3.3. Обзор литературы 33](#_Toc485340783)

[1.3.4. Подготовка инструментов 35](#_Toc485340784)

[2. Реализация 47](#_Toc485340785)

[2.1. Требования к приложению 47](#_Toc485340786)

[2.2. Архитектура 47](#_Toc485340787)

[2.2.1. Описание структуры плагина 47](#_Toc485340788)

[2.2.2. Описание структуры файла-дескриптора 47](#_Toc485340789)

[2.2.3. Описание структуры java класса реализатора 50](#_Toc485340790)

[2.2.4. Описание структуры velocity шаблона 51](#_Toc485340791)

[2.3. Плагины в «Atlassian JIRA» 52](#_Toc485340792)

[2.4. Технология доступа к внешним источникам данных 52](#_Toc485340793)

[2.5. Реализация плагина «Принятие решений» 54](#_Toc485340794)

[2.5.1. Создание каркаса плагина 54](#_Toc485340795)

[2.5.2. Расширение каркаса модулями 54](#_Toc485340796)

[2.5.3. Редактирование файла дескриптора 55](#_Toc485340797)

[2.5.4. Бизнес логика 58](#_Toc485340798)

[2.5.5. Написание java класса реализатора 60](#_Toc485340799)

[2.5.6. Написание velocity шаблона 63](#_Toc485340800)

[2.5.7. Использование JavaScript для вывода графиков 65](#_Toc485340801)

[2.6. Результаты 66](#_Toc485340802)

[Заключение 70](#_Toc485340803)

[Литература 71](#_Toc485340804)

[Приложение, содержание прилагаемого CD-диска 75](#_Toc485340805)

# Введение

В настоящее время проблема менеджмента своего рабочего времени является одной из преимущественно значимых задач в организации как учебного, так и многих других процессов, часто информация необходимая для принятия обоснованных и своевременных решений распределена либо не представлена в систематизированном виде, что осложняет ее анализ.

Существует ряд средств позволяющих решить данную проблему, таких как «Todoist» – мобильное приложение для организации своего списка дел или «Microsoft OneNote» - программа-органайзер, входящая в состав пакета «Microsoft Office», но все они имеют ряд недостатков, главным из которых является невозможность эффективно оценивать приоритетность задачи относительно других при иерархической структуре задач, когда у задачи есть ряд подзадач, у этих подзадач есть свои подзадачи и так далее. Данную проблему позволяет решить программная система управления проектами «JIRA».

На сегодняшний день «Atlassian JIRA» является одной из самых известных систем управления проектами. «JIRA» достаточно универсальна, чтобы решать большое число казалось бы несвязанных друг с другом задач. Однако в чистом виде «JIRA», как и другие подобные сервисы, не имеет необходимой функциональности для анализа больших и сложно структурированных проектов и принятия на его основе обоснованного решения, но предоставляет возможность для расширения системы посредством написания плагинов.

Целью данной работы является разработка плагина «Принятие решений» для программной системы управления проектами «JIRA», который бы позволил решить проблему менеджмента рабочего времени.

Для достижения данной цели были поставлены и решены следующие задачи:

1. Анализ существующих средств решения проблемы менеджмента рабочего времени.
2. Анализ и изучение системы управления проектами «JIRA».
3. Изучение и тестирование технологии плагинов «JIRA» с возможностью доступа к внешним источникам данных.
4. Литературный обзор информации и технологий необходимых для решения проблемы.
5. Разработка критерия принятия решений.
6. Проектирование и разработка плагина «Принятие решений» для программной системы управления проектами «JIRA».

# Обзор литературы и подготовка инструментария

## Обзор существующих решений

В ходе выполнения данной работы был проведен обзор, тестирование и анализ ряда программных продуктов, решающих проблему менеджмента рабочего времени, далее будет приведены описание и результаты данного анализа, которые были отражены в виде таблицы оценок по 10-бальной шкале, которые демонстрируют на сколько то или иное программное обеспечение решает поставленные проблемы:

* Проблема менеджмента рабочего времени;
* Проблема анализа большого количества задач;
* Проблема анализа иерархической структуры задач;
* Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени.

### «Todoist»

«Todoist» — это кроссплатформенный сервис, доступный для мобильных устройств таких как «iOS», «Android», «Windows Phone», а также «Symbian». Кроме того существуют расширения для «Chrome» и «Firefox». Чтобы воспользоваться данным сервисом необходимо пройти простую регистрацию, либо при наличии «Google» аккаунта использовать его для входа. После аутентификации необходимо создать проект, в котором затем нужно внести список задач, расставить приоритеты, а также определить сроки их выполнения. На этом функциональность бесплатной версии данного продукта заканчивается, но существует так же и платная версия «Todoist», которая обойдется пользователю в 29 долларов в год или 2 доллара в месяц. Обладатели платной версии помимо базовых возможностей так же могут просматривать решенные задачи и отслеживать статистику своей продуктивности, делиться списками дел, добавлять комментарии к задачам, получать уведомления по электронной почте или SMS, а также экспортировать задачи в календарь «Google», «Outlook» или «iCal». Подробнее о «Todoist», можно узнать на официальном сайте проекта[1]. На Рис. 1 изображен пример отображения задач в виде плана на ближайшие 7 дней в сервисе «Todoist».

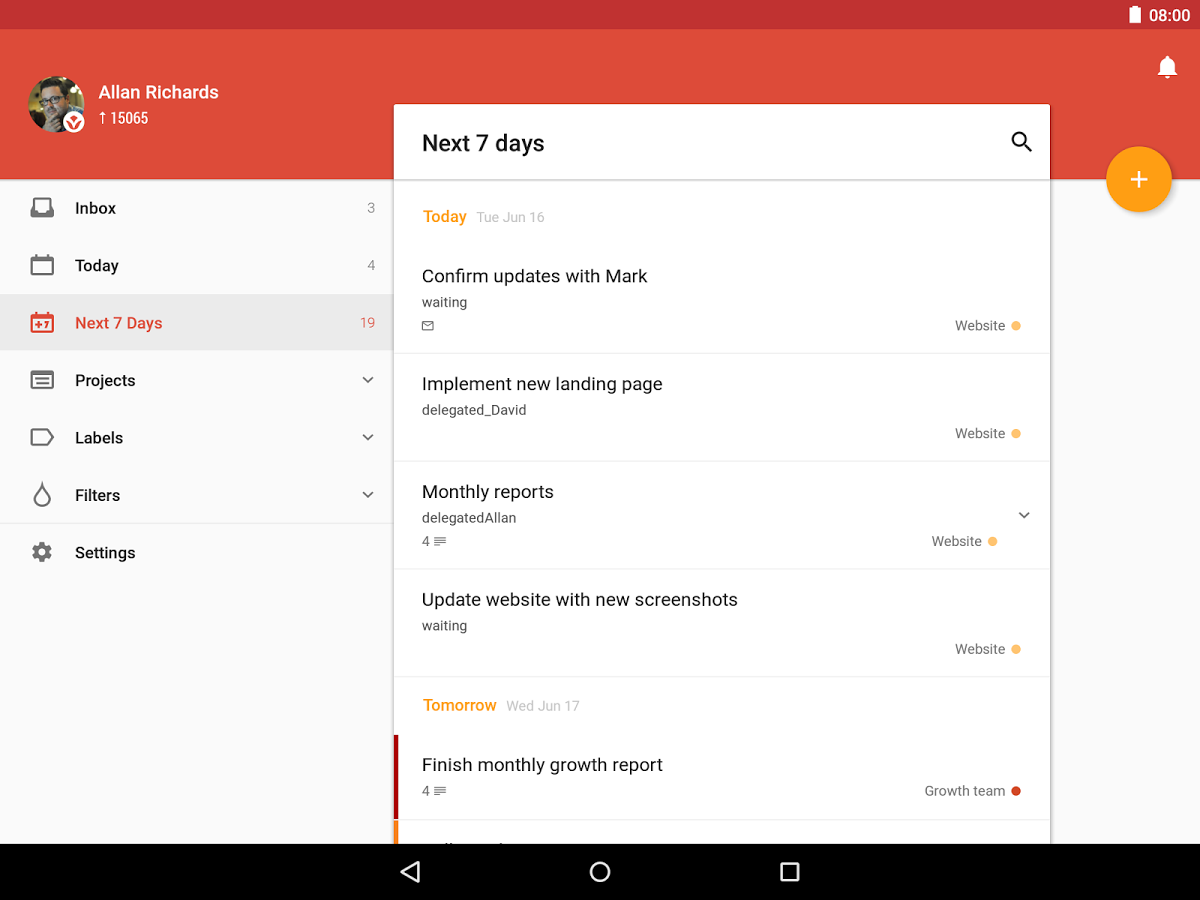


Рис. 1 План на 7 дней, «Todoist»

Результаты анализа платной и бесплатной версии «Todoist» приведены в Таблице 1.

|  |  |  |
| --- | --- | --- |
| Проблемы ↓ **\** Версия → | Платная | Бесплатная |
| Проблема менеджмента рабочего времени. | 7/10 | 6/10 |
| Проблема анализа большого количества задач. | 8/10 | 7/10 |
| Проблема анализа иерархической структуры задач. | 6/10 | 6/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 4/10 | 4/10 |

Таблица 1, результаты анализа приложения «Todoist»

### «Any.do»

Следующим рассматриваемым менеджером задач стал сервис «Any.do». Рабочий экран приложения разделен на четыре раздела: «Сегодня», «Завтра», «Позже» и «Без времени». Добавить задачи в любой из разделов можно не только вручную, но и с помощью голоса. Помимо задач сервис «Any.do» позволяет создавать два типа уведомлений. Первый тип — уведомления, которые срабатывают в указанное пользователем время, второй — напоминания, которые срабатывают по прибытии в определенную точку на карте. Так же в «Any.do» есть возможность добавлять заметки к делам и определять их приоритетность. Благодаря функции «Moment» можно быстро просмотреть свои первоочередные задачи. «Any.do» работает на «iOS», «Android» и «Google Chrome». Данный сервис бесплатен. Подробнее о «Any.do», можно узнать на официальном сайте проекта[2]. На Рис. 2 изображен рабочий экран приложения «Any.do» с рядом задач.

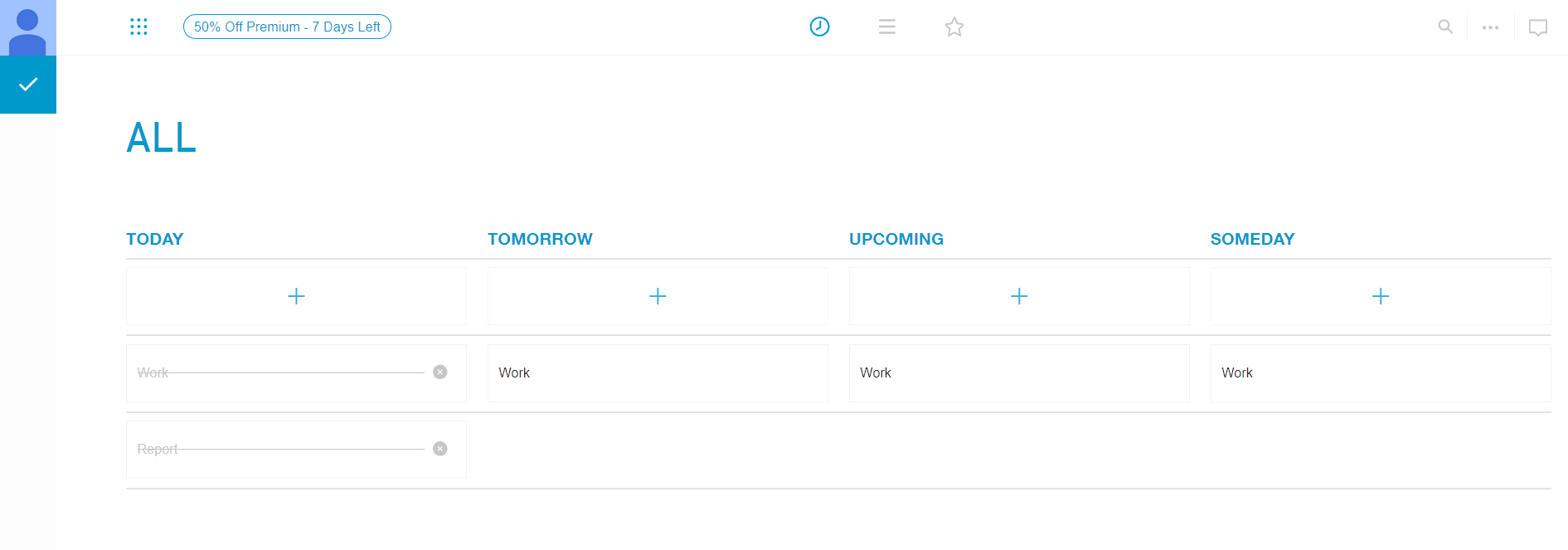


Рис. 2 Рабочий экран «Any.do»

Результаты анализа приложения «Any.do» приведены в Таблице 2.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 4/10 |
| Проблема анализа большого количества задач. | 5/10 |
| Проблема анализа иерархической структуры задач. | 2/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 2/10 |

Таблица 2, результаты анализа приложения «Any.do»

### «GanttPro»

«GanttPro» – сервис позволяющий строить диаграмму «Гантта», а так же планировать и управлять проектами онлайн, кроме того данное программное обеспечение дает возможность визуализировать процессы, создавать задачи и назначать их участникам, выставлять сроки завершения задач и процент завершения отдельных задач, а так же проекта в целом, добавлять вехи, и делиться созданным графиком «Гантта» с командой и с клиентами с правом просмотра или редактирования.

Плюсы «GanttPro»:

* Автоматический процесс планирования проектов.
* Возможность добавления файлов к существующим диаграммам.
* Возможность выставления процента выполнения задачи и проекта
* Возможность выставлять собственные рабочие дни и часы.
* Автоматический пересчет длительностей задач и проектов.
* Выставление зависимостей между задачами.
* Возможность создания сложных, многоуровневых проектов.

Минусы «GanttPro»:

* Нет возможности выставить приоритеты задач.
* Отсутствуют уведомления.

Подробнее о «GanttPro», можно узнать на официальном сайте проекта[3]. На Рис. 3 изображен рабочий экран сервиса «GanttPro» с построенной диаграммой «Гантта».

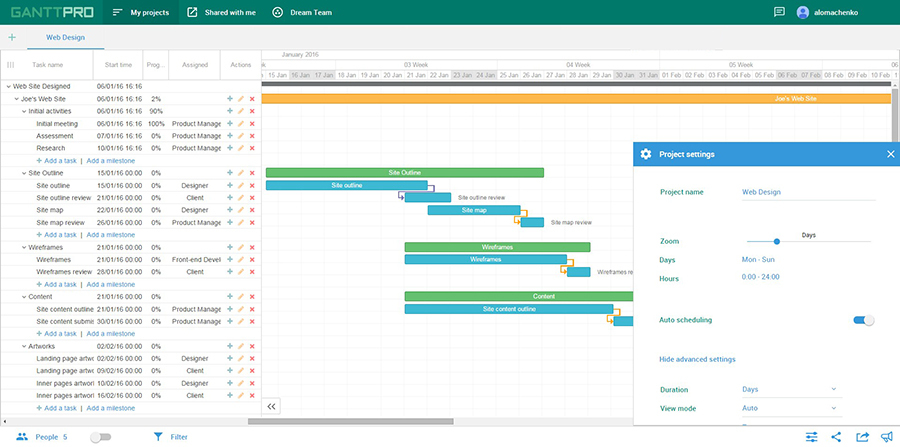


Рис. 3 Рабочий экран «GanttPro»

Результаты анализа приложения «GanttPro» приведены в Таблице 3.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 8/10 |
| Проблема анализа большого количества задач. | 7/10 |
| Проблема анализа иерархической структуры задач. | 10/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 6/10 |

Таблица 3, результаты анализа приложения «GanttPro»

### «Asana»

«Asana» – это web-приложение, но также существуют клиенты для «iOS» и «Android», которое позволяет назначать задачи, выставлять приоритеты и сроки выполнения, а так же следить за статусом выполнения задач и проекта в целом.

Плюсы «Asana»:

* Данный сервис прост в использовании.
* Существует возможность управлять несколькими проектами в рамках одной команды.
* Система тэгов для задач.
* Система анализа эффективности.

Минусы «Asana»:

* Нет возможности назначать одну задачу группе работников.
* Нет возможности оценить проект в целом.

Подробнее о «Asana», можно узнать на официальном сайте проекта[4]. На Рис. 4 изображен рабочий экран сервиса «Asana» с открытой страницей анализа проектов.

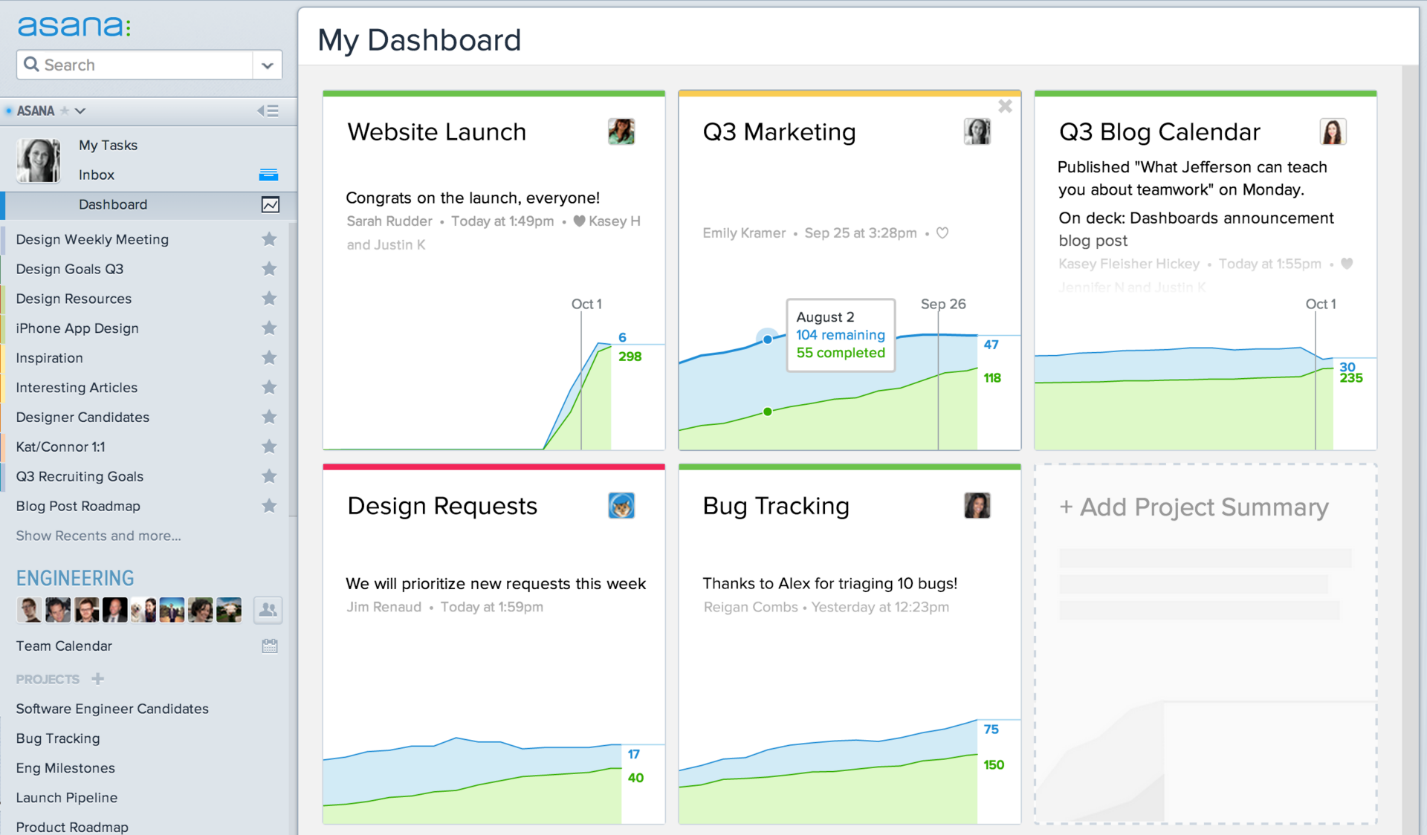


Рис. 4 Рабочий экран «Assana»

Результаты анализа приложения «Assana» приведены в Таблице 4.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 8/10 |
| Проблема анализа большого количества задач. | 8/10 |
| Проблема анализа иерархической структуры задач. | 7/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 7/10 |

Таблица 4, результаты анализа приложения «Assana»

### «Easy Redmine»

«Easy Redmine» – это «Open Source» софт для управления задачами, предназначен для планирования и управления большими проектами с возможностью расширения для управления ресурсами, финансами, «CRM» и многим другим.

Плюсы «Easy Redmine»:

* Можно установить сервис на свой локальный сервер.
* Хорошо подходит для больших команд.

Минусы «Easy Redmine»:

* Нет общей визуализации проектов.
* Неудобная навигация.
* Нет возможности быстро планировать спринты.

Подробнее о «Easy Redmine», можно узнать на официальном сайте проекта[5]. На Рис. 5 изображен рабочий экран приложения «Easy Redmine», демонстрирующий информацию по проектам.

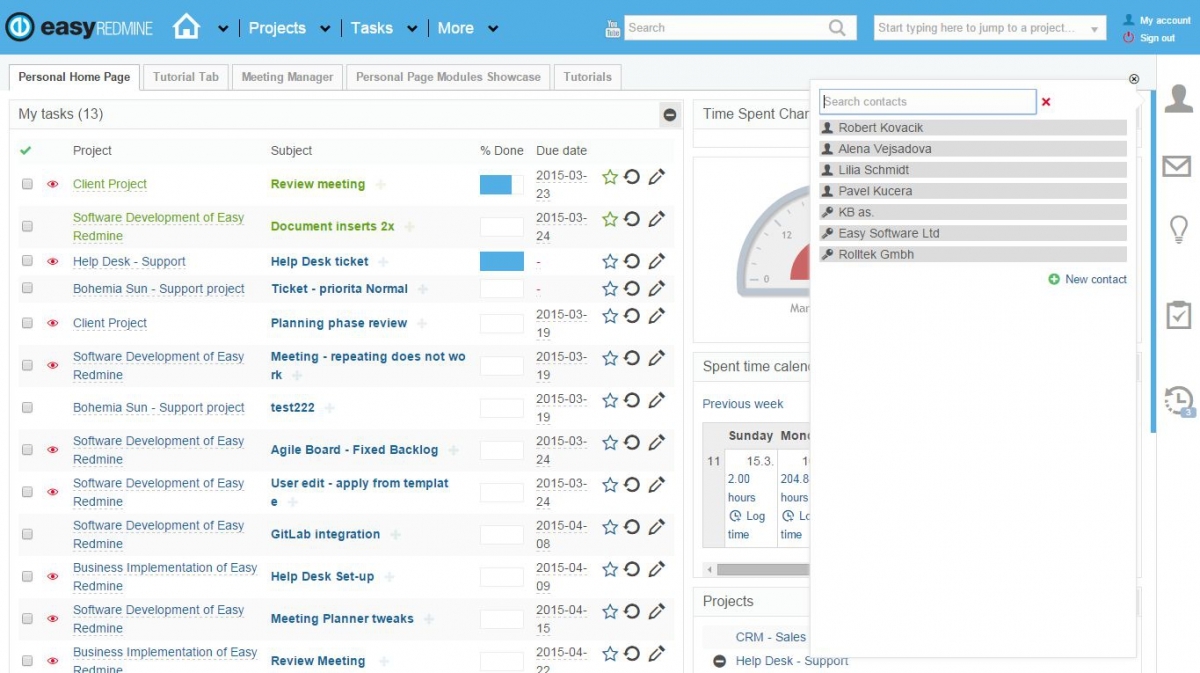


Рис. 5 Рабочий экран «Easy Redmine»

Результаты анализа приложения «Easy Redmine» приведены в Таблице 5.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 7/10 |
| Проблема анализа большого количества задач. | 9/10 |
| Проблема анализа иерархической структуры задач. | 7/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 8/10 |

Таблица 5, результаты анализа приложения «Easy Redmine»

### «Freedcamp»

«Freedcamp» – это сервис представляет из себя организованное рабочее пространство для мгновенного просмотра целого проекта, а так же выставления и назначения задач, используя «стикеры» и календарь. Кроме того «Freedcamp» предлагает дополнения для CRM, составления инвойсов, баг-трекинга, а так же создания wiki страниц.

Плюсы «Freedcamp»:

* Существует возможность выставить приоритеты задачам.
* Есть возможность разделить все проекты на отдельные вкладки.
* Прост в использовании.

Минусы «Freedcamp»:

* Нельзя выставить выставить прогресс и процент выполнения задач.

Подробнее о «Freedcamp», можно узнать на официальном сайте проекта[6]. На Рис. 6 изображен рабочий экран сервиса «Freedcamp», демонстрирующий этапы или вехи проекта.

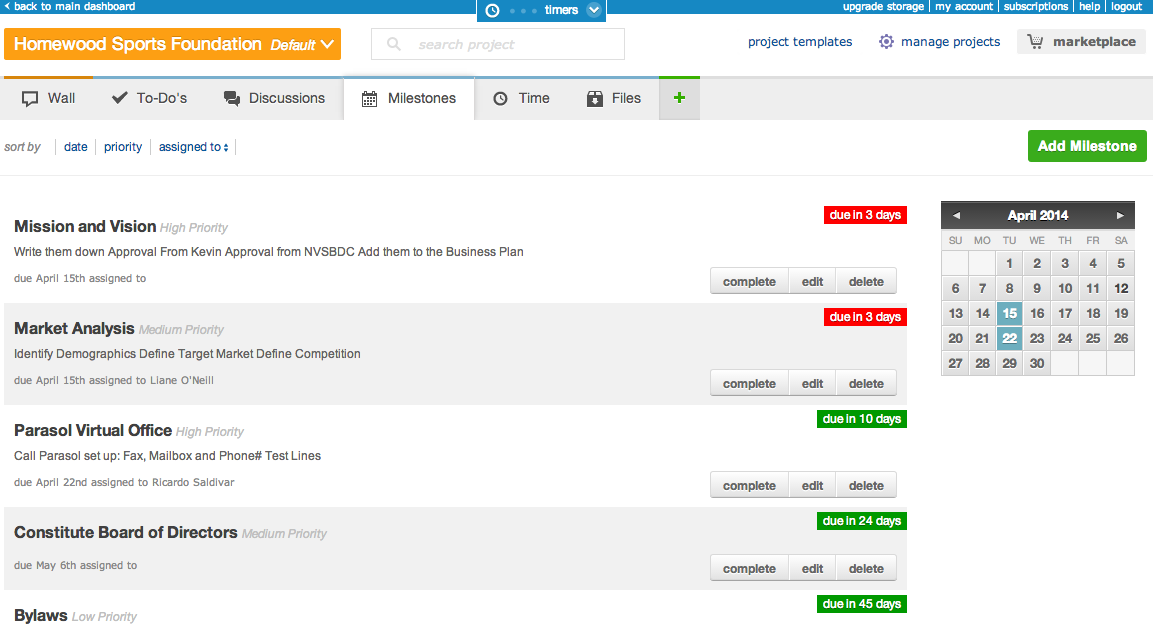


Рис. 6 Рабочий экран «Freedcamp»

Результаты анализа приложения «Freedcamp» приведены в Таблице 6.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 8/10 |
| Проблема анализа большого количества задач. | 8/10 |
| Проблема анализа иерархической структуры задач. | 6/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 8/10 |

Таблица 6, результаты анализа приложения «Freedcamp»

### «Bitrix24»

«Bitrix24» – это отечественный сервис, включающий в себя подсистему управления задачами, планирование проектов, а так же времени их выполнения, визуализацию информации на календаре, возможность делиться документами, кроме того «Bitrix24» позволяет создавать виртуальные рабочие группы и получать уведомления.

Плюсы «Bitrix24»:

* Широкая область применения.

Минусы «Bitrix24»:

* Тяжел в использовании.
* Минимальная визуализация задач и проектов в целом.
* Невозможно изменить интерфейс.
* Огромное количество особенностей, которые никогда не понадобятся.
* Ценовая политика взыскания дополнительной платы за обновления.

Подробнее о «Bitrix24», можно узнать на официальном сайте проекта[7]. На Рис. 7 изображен рабочий экран сервиса «Bitrix24», демонстрирующий «Живую ленту» проекта.

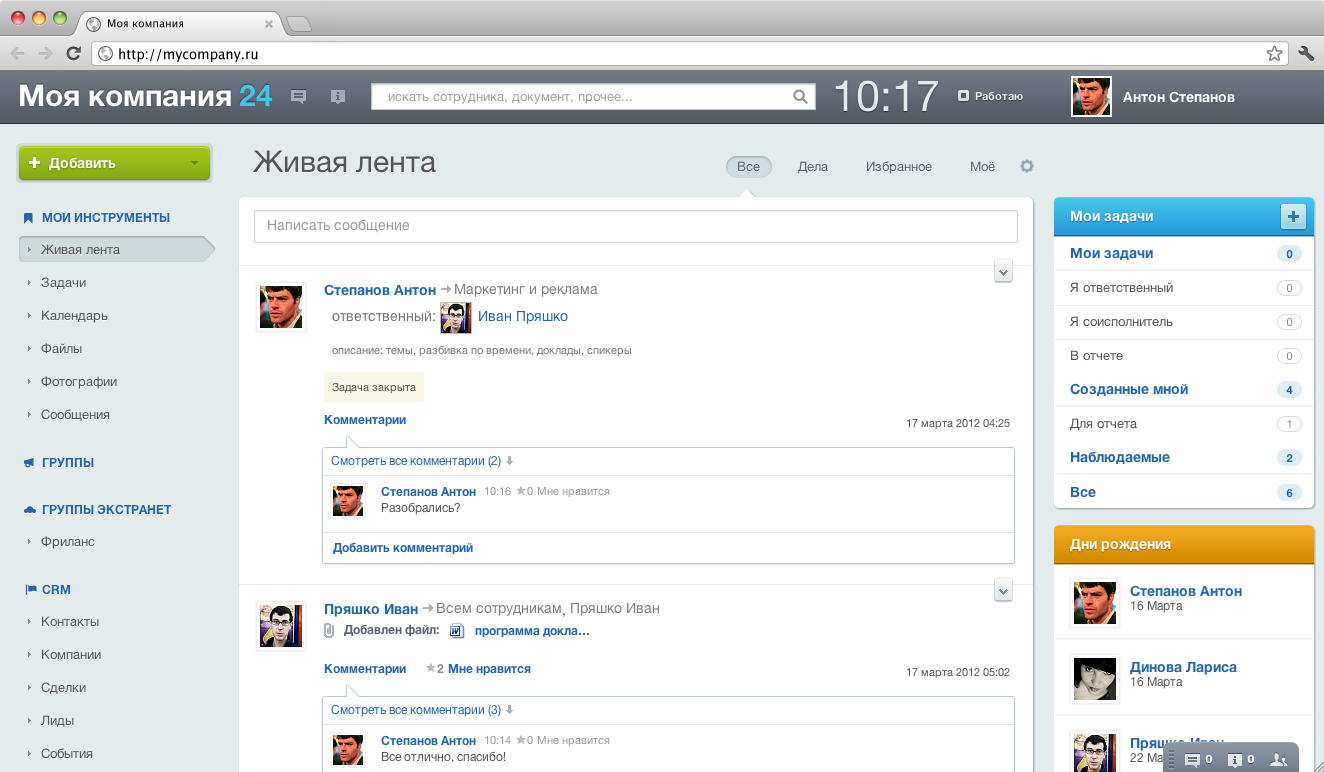


Рис. 7 Рабочий экран «Bitrix24»

Результаты анализа приложения «Bitrix24» приведены в Таблице 7.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 6/10 |
| Проблема анализа большого количества задач. | 6/10 |
| Проблема анализа иерархической структуры задач. | 5/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 5/10 |

Таблица 7, результаты анализа приложения «Bitrix24»

### «Podio»

«Podio» – это сервис, позволяющий работать над проектами вместе с командой, а так же управлять проектами и задачами и хранить и обмениваться файлами. Приложение так же можно использовать как CRM систему, или же интранет, для взаимодействия между работниками одной компании.

Плюсы «Podio»:

* Гибкость и возможность настраивать сервис под себя.
* Возможность назначать ресурсы, а так же следить за различными данными, такими как бюджет или расходы.

Минусы «Podio»:

* Неудобная навигация.

Подробнее о «Podio», можно узнать на официальном сайте проекта[8]. На Рис. 8 изображен рабочий экран сервиса «Podio», демонстрирующий экран топологических схем проекта.

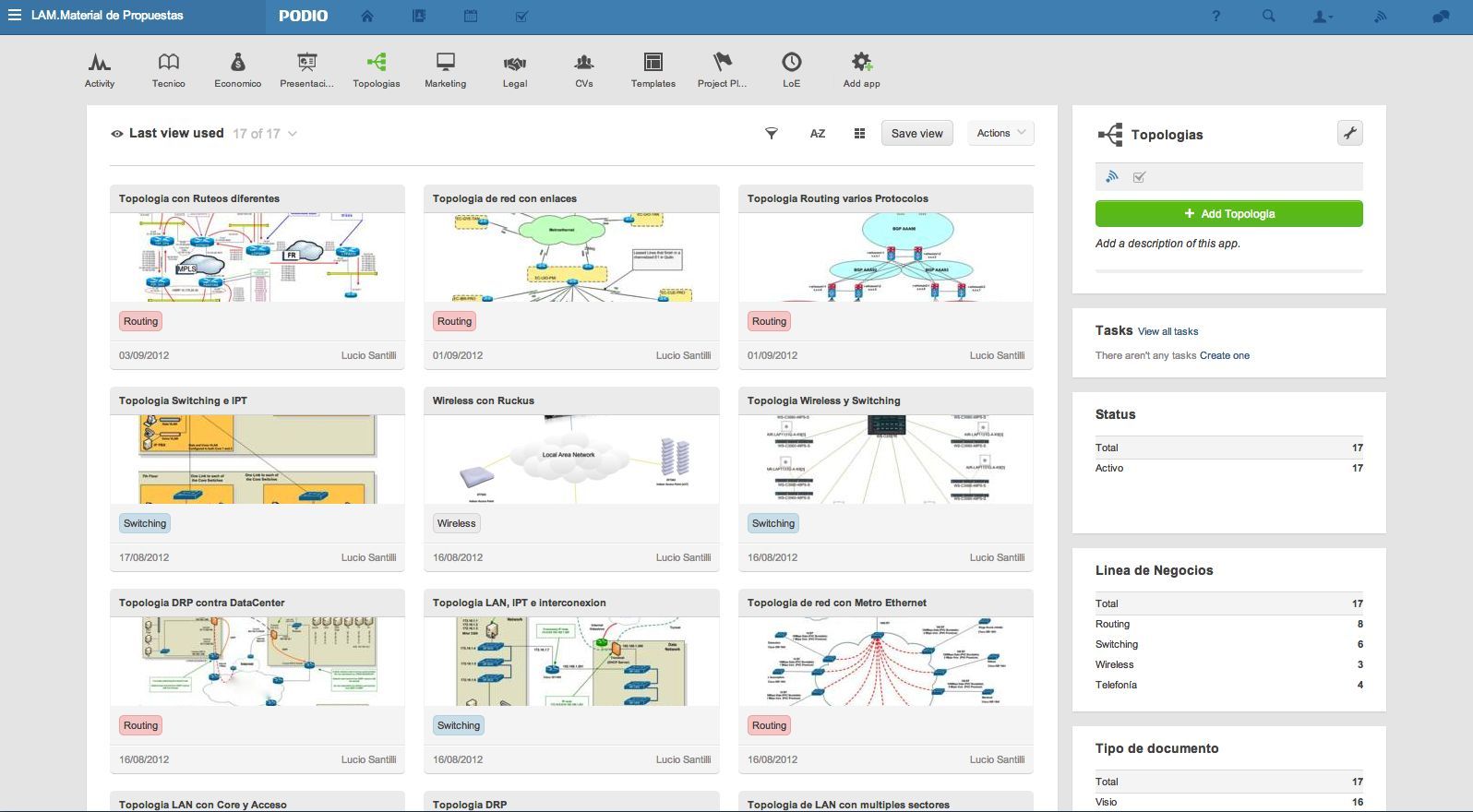


Рис. 8 Рабочий экран «Podio»

Результаты анализа приложения «Podio» приведены в Таблице 8.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 8/10 |
| Проблема анализа большого количества задач. | 8/10 |
| Проблема анализа иерархической структуры задач. | 7/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 6/10 |

Таблица 8, результаты анализа приложения «Podio»

### «Wrike»

«Wrike» – это единая система для работы над проектами вместе с командой с расширенными функциональными особенностями: позволяет назначать роли, а так же создавать рабочее пространство для каждого отдела, ставить и обсуждать задачи, следить за их статусом и прогрессом, делиться файлами и отображать задачи проекта на диаграмме «Гантта», и получать еженедельные отчеты.

Плюсы «Wrike»:

* Сервис удобен как для больших, так и для маленьких команд.
* Существует возможность создавать отдельные папки для отдельных проектов.

Минусы «Wrike»:

* Неудобная навигация в многоуровневых проектах.

Подробнее о «Wrike», можно узнать на официальном сайте проекта[9]. На Рис. 9 изображен рабочий экран сервиса «Wrike», демонстрирующий информацию по текущему проекту.

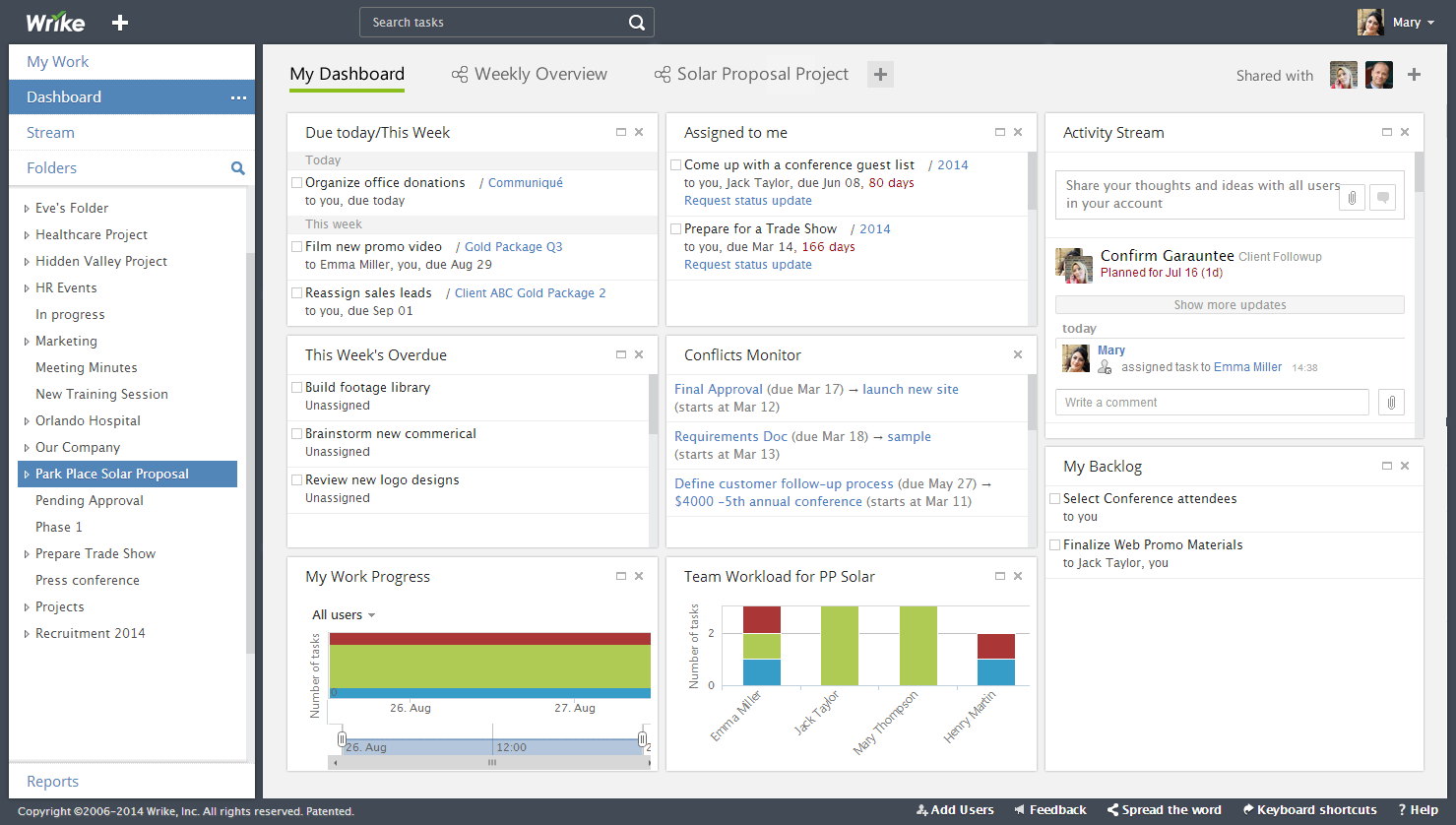


Рис. 9 Рабочий экран «Wrike»

Результаты анализа приложения «Wrike» приведены в Таблице 9.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 9/10 |
| Проблема анализа большого количества задач. | 9/10 |
| Проблема анализа иерархической структуры задач. | 6/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 7/10 |

Таблица 9, результаты анализа приложения «Wrike»

### 1.1.10. «Jira»

«Jira» – это мощный онлайн сервис, но так же существует и версия для компьютеров, которая позволяет поднять свой собственный сервер «Jira». Данное программное обеспечение позволяет командам-разработчикам планировать проекты, назначать исполнителей задач, планировать рабочие процессы, а так же собирать задачи в бэклог, выставлять приоритеты и сроки выполнения задач. Кроме того «Jira» позволяет достаточно просто расширять свою функциональность с помощью системы плагинов, что позволяет адаптировать эту систему под любые нужды.

Плюсы «Jira»:

* Большое количество настроек фильтрации.
* Продуманная система визуализации с возможностью изменения.
* Возможность выбора причин для закрытия задач.
* Детальные и проработанные отчеты.
* Простота в управлении проектами.
* Простой и удобный трэкер времени.
* Хорошие возможности для расширения.

Минусы «Jira»:

* Необходимо долгое обучение перед началом эффективной работы.
* Недостаточно проработанная система поддержки пользователей.

На Рис. 10 изображен рабочий экран сервиса «Jira», демонстрирующий информацию по текущему проекту.

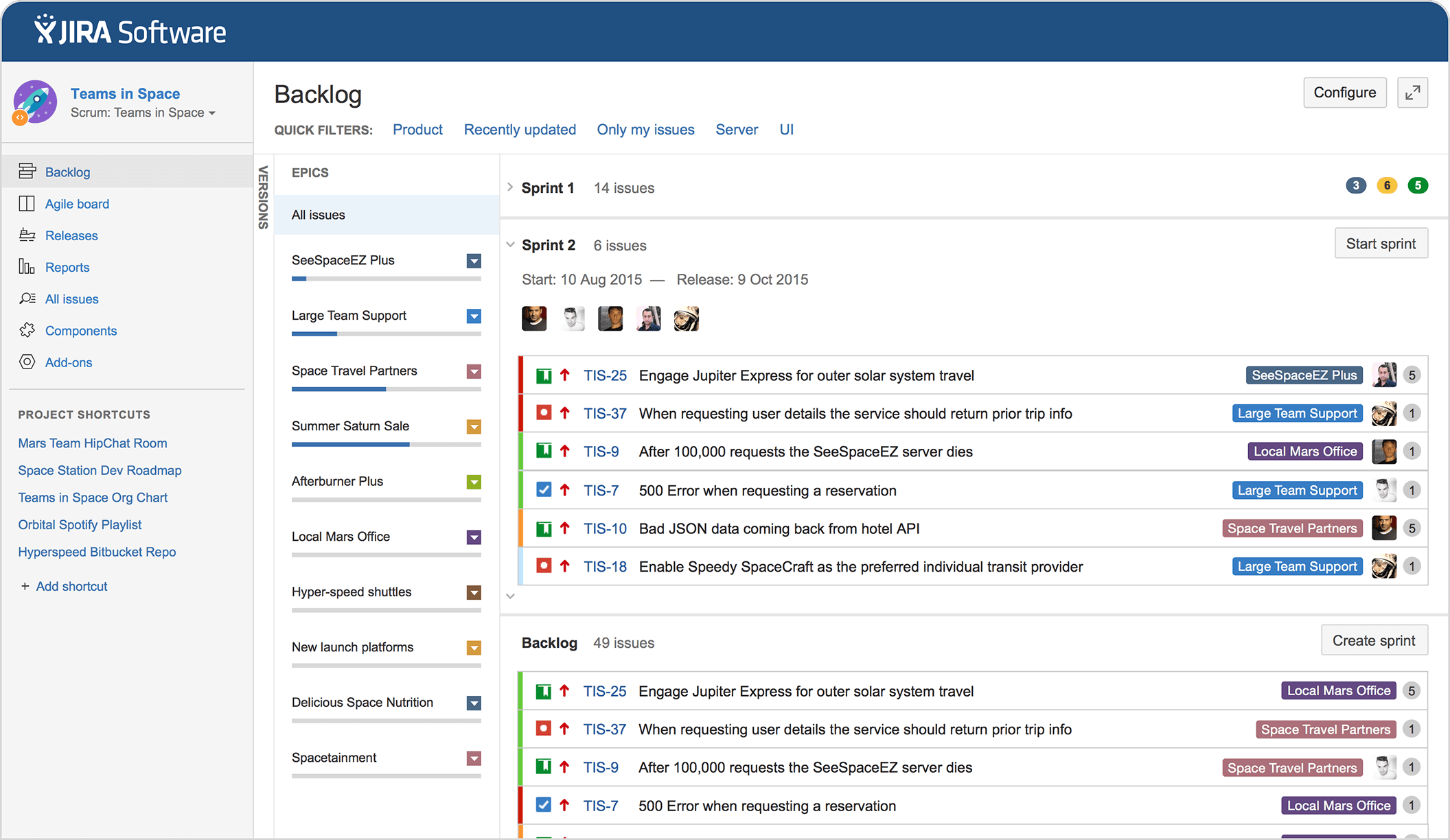


Рис. 10 Рабочий экран «Jira»

Результаты анализа приложения «Jira» приведены в Таблице 10.

|  |  |
| --- | --- |
| Проблемы | Оценка |
| Проблема менеджмента рабочего времени. | 9/10 |
| Проблема анализа большого количества задач. | 9/10 |
| Проблема анализа иерархической структуры задач. | 9/10 |
| Проблема выработки единого критерия для оценки важности задачи в конкретный момент времени. | 8/10 |

Таблица 10, результаты анализа приложения «Jira»

В данной работе за основу была выбрана именно «Jira» из-за того, что у нее наивысшая совокупная оценка, а так же она сравнительно легко расширяема с помощью плагинов.

## Обзор способов отображения информации по проекту

Помимо исследования существующих способов решить проблему менеджмента рабочего времени, было проведено исследование существующих способов отображения информации по проекту на предмет их плюсов и минусов при различных условиях, всего было выделено и исследовано 3 основных вида графического представления информации, подходящих для данной задачи, это: диаграмма «Гантта», ментальные карты и деревья. Далее будет приведено описание каждого из рассмотренных способов.

### Диаграмма «Гантта»

Диаграмма «Гантта» — это популярный тип гистограмм, используемый для демонстрации в графическом виде различных планов и графиков работ по какому-либо проекту. Является одним из самых распространенных методов планирования проектов, а так же используется в многих приложениях по управлению проектами.

Диаграмма «Гантта» состоит из полос, расположенных вдоль оси времени. Каждая полоса на диаграмме представляет отдельную задачу в составе проекта, а её концы — моменты начала и завершения работы, её протяженность — длительность работы. Вертикальной осью диаграммы служит перечень задач. Кроме того, на диаграмме могут быть отмечены совокупные задачи, проценты завершения, указатели последовательности и зависимости работ, а так же метки ключевых моментов «вехи».

Одним из главных понятий диаграммы «Гантта» является «Веха», которая представляет из себя метку значимого момента в ходе выполнения работ, общую границу двух или более задач. «Вехи» позволяют наглядно отобразить необходимость синхронизации, последовательности в выполнении различных работ. Сдвиг вехи приводит к сдвигу всего проекта. Диаграмма «Гантта» не отображает приоритетности или сложности работ, а так же не отображает сущности работ. Для крупных проектов диаграмма «Гантта» становится чрезмерно тяжеловесной и теряет всякую наглядность. На Рис. 11 изображен пример диаграммы «Гантта», демонстрирующий 4 задачи, которые необходимо выполнить за Февраль.[10]

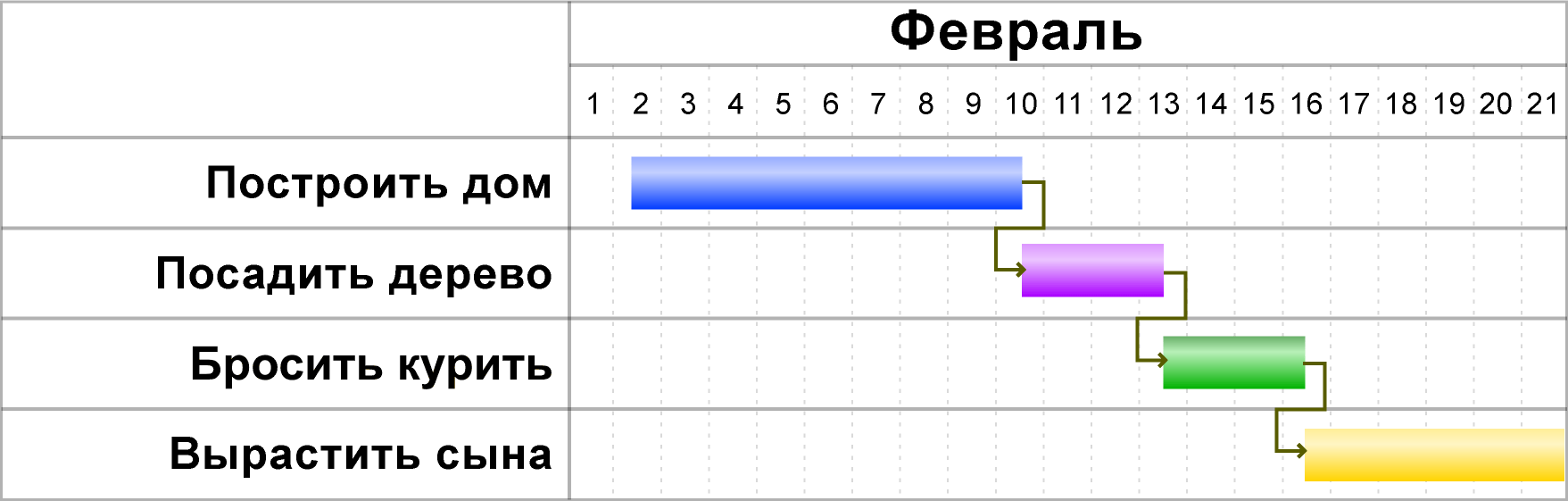


Рис. 11, пример диаграммы «Гантта»

### Ментальная карта

Ментальные карты — это техника визуализации мышления. Применения ментальных карт очень разнообразны — например, их можно использовать для того, чтобы зафиксировать, понять и запомнить содержание книги или текста, сгенерировать и записать идеи, разобраться в новой для себя теме, подготовиться к принятию решения, кроме того ментальные карты активно используются для визуализации различных проектов.

Ментальная карта реализуется в виде древовидной схемы, на которой изображены слова, идеи, задачи или другие понятия, связанные ветвями, отходящими от центрального понятия или идеи. На Рис. 12 приведен пример такой карты для задачи постройки дома. Достоинством данного метода отображения информации является структурированность и легкость получения общего представления о заложенных в карту знаниях. К недостаткам следует отнести ограниченную масштабируемость и сфокусированность на единственном центральном понятии.[11]

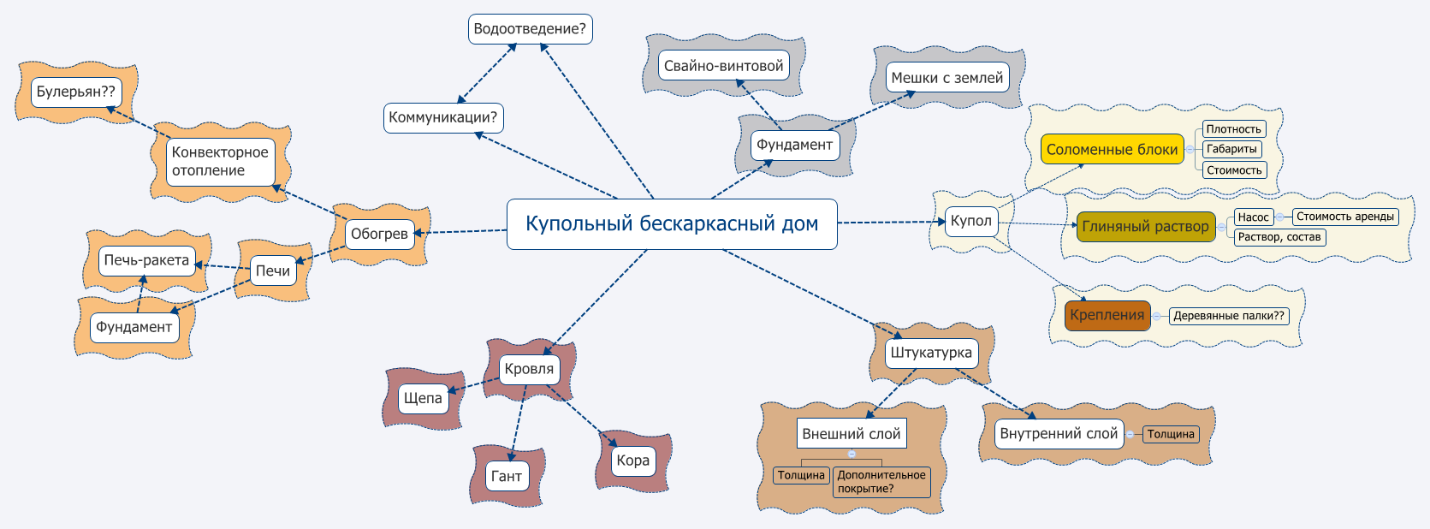


Рис. 12, пример ментальной карты

### Деревья

Дерево — это связный ациклический граф, связность означает наличие путей между любой парой вершин, ацикличность — отсутствие циклов и то, что между парами вершин имеется только по одному пути. Лес — упорядоченное множество упорядоченных деревьев.[12] В случаи системы управления проектами «JIRA», лес представляет из себя множество всех задач проект(а/ов), у которых нет задач родителей, каждая такая задача представляет из себя дерево, где ветвями являются подзадачи данной задачи, у которых в свою очередь могут так же быть свои подзадачи-ветви. Древовидный способ отображения информации по проекту является наиболее подходящим для задач со сложной иерархической структурой, а так же больших проектов, включающих в себя огромное количество задач, именно поэтому данный вид отображения информации по проекту и был выбран за основу при реализации данной работы. На Рис. 13 изображен пример дерева, отображающего информацию по текущему проекту.

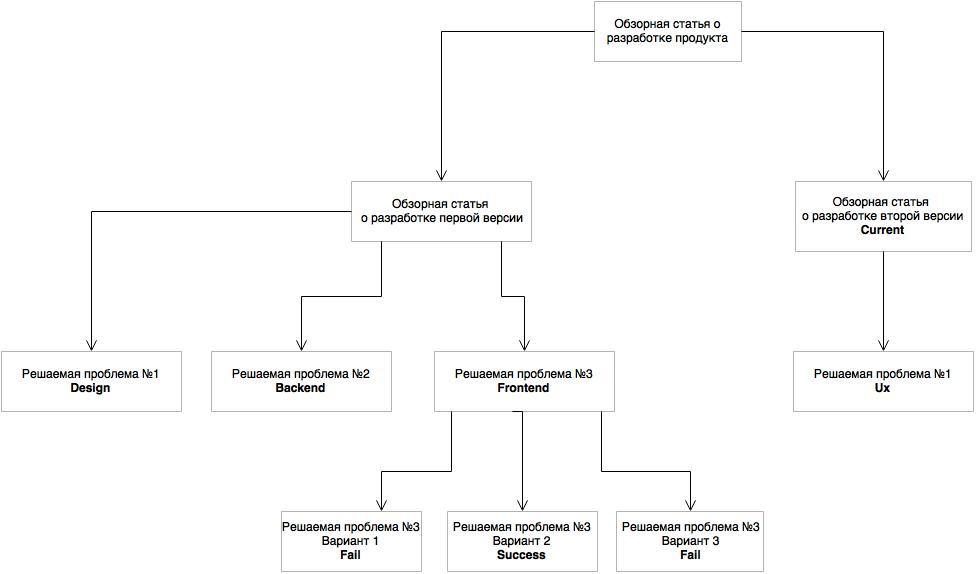


Рис. 13, пример дерева

## Обзор литературы по теме и подготовка инструментов

### Сущности project, issue и связи между ними

В системе управления проектами «JIRA» для представления проектов и задач используются сущности project и issue.

«JIRA» project это коллекция issue, он определяется в соответствии с требованиями организации, например project может быть:

* Проект разработки программного обеспечения;
* Проект маркетинговый компании;
* Проект по внедрению технологии;
* Система справочной службы;
* Система управления запросами на отпуск;
* Система запроса на повышение качества сайта.

Каждая issue принадлежит проекту. Каждый project имеет свое имя (например WebSite issues), а так же ключ (например WEB). Ключ project становится первой частью ключа issue, например WEB-101, WEB-102 и так далее[13].

Различные организации используют «JIRA» для отслеживания различных проблем. В зависимости от варианта использования системы issue, например может быть:

* Ошибкой программного обеспечения;
* Задачей проекта;
* Отчетом службы поддержки;
* Формой запроса на отпуск.

На Рис. 14 представлен пример issue.

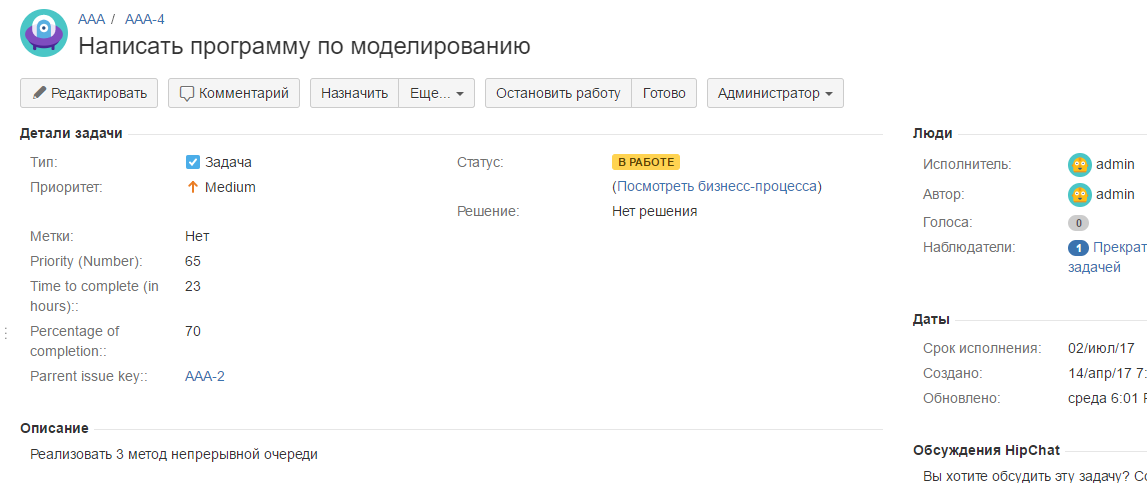


Рис 14. Пример issue.

Основные поля, которые содержатся в issue[14]:

* Project – имя проекта, к которому принадлежит issue, например WebSite issues;
* Key – уникальный идентификатор текущей issue, например WEB-101;
* Summary – краткое описание в 1 строчку текущей issue, например «Оплатить хостинг»;
* Type – тип issue, существует всего 4 основных типа issue: Bug, Improvement, New Feature, Task, а так же пользовательские типы, которые нужно создавать самостоятельно;
* Status – текущая стадия issue в рабочем процессе, всего существует 5 статусов: Open, In Progress, Resolved, Reopened и Closed;
* Priority – приоритет текущей issue относительно других issue, существует всего 5 уровней приоритета: Very High, High. Medium, Low и Very Low;
* Description – подробное описание текущей issue;
* Assignee – поле хранящие имя пользователя, который отвечает за выполнение данной issue;
* Due – дата до которой данная issue должна быть решена.

Так же система управления проектами «JIRA» позволяет создавать пользовательские поля и привязывать их к issue. Для решения проблемы менеджмента рабочего времени в данной работе были созданы поля:

* Priority (Number) - приоритет текущей issue относительно других issue, в отличии от стандартного поля Priority хранит произвольное число, которое в дальнейшем будет нормировано и использовано в расчетах;
* Time to complete (in hours) – время необходимое на выполнение задачи, выраженное в часах, необходимое для расчета параметра сложности задачи;
* Percentage of completion – процент выполнения текущей issue, необходимо для расчета критерия принятия решений.

### «Atlassian JIRA»: обзор, реализация, лицензия, архитектура и решаемые проблемы

**Обзор**

«Atlassian JIRA» — это коммерческая система отслеживания ошибок, предназначенная для организации взаимодействия с пользователями, так же активно используется и как система управления проектами. Разработана компанией «Atlassian», является одним из двух её основных продуктов (наряду с вики-системой «Confluence»). Имеет веб-интерфейс.

Название системы получено путём усечения слова «Gojira» — японского имени монстра Годзилла, что, в свою очередь, является отсылкой к названию конкурирующего продукта — «Bugzilla»; создавалась в качестве замены «Bugzilla» и во многом повторяет её архитектуру. Система позволяет работать с несколькими проектами. Для каждого из проектов создаёт и ведёт схемы безопасности и схемы оповещения[15].

**Решаемые проблемы**

Основные проблемы решаемые программной системой управления проектами «JIRA»:

* Планирования – позволяет создавать пользовательские задачи, планировать рабочие процессы, а так же распределять задачи среди пользователей системы;
* Отслеживания – позволяет расставлять приоритеты задачам, а так же обсуждать работу внутри команды;
* Выпуска новых версий – позволяет выпускать новые версии разрабатываемого продукта предоставляя самую актуальную информацию в удобном виде;
* Создания отчетов – позволяет повысить работоспособность путем предоставления отчетов о текущей работе в реальном времени[16].

**Реализация**

Система управления проектами «JIRA» основана на «Java EE» и работает на многих популярных системах управления базами данных и операционных системах.

Основной элемент учёта в системе — это задача (issue). Задача содержит название проекта, тему, тип, приоритет, компоненты и содержание. Задача может быть расширена дополнительными полями. Задача может редактироваться или изменять статус. Любые изменения протоколируются в журнал.

«JIRA» имеет большое количество возможностей конфигурации: для каждого приложения может быть определен отдельный тип задачи с собственным рабочим процессом, набором статусов, одним или несколькими видами представления. С помощью схем можно определить для каждого индивидуального проекта собственные права доступа, поведение и видимость полей.

Благодаря универсальному подходу можно приспособить «JIRA» для многих непрофильных задач, например, управления требованиями, управления рисками.

Для интеграции с внешними системами поддерживает интерфейсы SOAP, XML-RPC и REST. Поставляется со средствами интеграции с такими системами управления версиями, как «Subversion», «CVS», «Git», «Clearcase», «Team Foundation Server», «Mercurial» и «Perforce». Существуют дополнения, позволяющие встроить «JIRA» в интегрированные среды разработки, в том числе «Eclipse» и «IntelliJ IDEA».

Для сторонних разработчиков предоставляются средства разработки расширений системы — плагинов. Разработчики расширений могут выкладывать плагины для продажи на специальный раздел сайта «Atlassian» [15].

**Лицензия**

«Atlassian JIRA» является коммерческим продуктом, который может быть лицензирован для работы на локальном сервере или доступен в качестве удалённого приложения. Ценообразование зависит от максимального числа пользователей, например для небольшой команды до 10 пользователей «JIRA» обойдется в 10$ в месяц, для 100 пользователей 450$, а для 2000 пользователей уже 1500$. Для академических и коммерческих клиентов доступен полный исходный код под лицензией разработчика[15][16].

Для проектов с открытым исходным кодом «Atlassian» предоставляет специальную бесплатную лицензию при соблюдении следующих правил:

* + Проект использует лицензии, одобренные «Open Source Initiative»;
  + Исходный код проекта доступен для скачивания;
  + У проекта есть публично доступный вебсайт;
  + Программное обеспечение от «Atlassian» доступно на вебсайте проекта.

**Архитектура**

**WebWork**

Поскольку «JIRA» - это веб-приложение, пользователи взаимодействуют с «JIRA» с помощью веб-браузера. «JIRA» использует «OpenSymphony's WebWork 1» для обработки веб-запросов, отправленных пользователями. «WebWork 1» - это MVC-структура, подобная «Struts». Каждый запрос обрабатывается действием «WebWork», которое обычно использует другие объекты, такие как утилиты и классы «Manager» для выполнения задачи.

«JIRA» использует «JSP» для слоя отображения. Таким образом, большая часть HTML-кода, который подается пользователю в качестве ответа на запрос вебсайта, генерируется «JSP». Поэтому для генерации ответа действие «WebWork» использует «JSP»[17].

**Seraph**

Почти вся аутентификация в «JIRA» выполняется через «Seraph», платформу аутентификации «Atlassian» с открытым исходным кодом. Цель «Seraph» - предоставить простую расширяемую систему аутентификации, которую можно использовать на любом сервере приложений.

«Seraph» реализован как фильтр сервлета. Его единственная задача - с помощью веб-запроса - связать этот запрос с конкретным пользователем. Он поддерживает несколько методов аутентификации, включая базовую аутентификацию HTTP, проверку подлинности на основе форм (например, перенаправление на внутреннюю или внешнюю форму входа в систему) и поиск учетных данных, уже сохраненных в сеансе пользователя (например, куки).

«Seraph» сам не выполняет управление пользователями. Он просто проверяет учетные данные входящего запроса и делегирует любые функции управления пользователями (поиск пользователя, проверка правильности пароля пользователя) в пользовательское управление «JIRA» - «Embedded Crowd» [17].

**Embedded Crowd**

«Embedded Crowd» - это инструмент управления идентификацией Atlassian. «Embedded Crowd» предоставляет следующие функции:

* Хранит пользователей и группы в базе данных «JIRA»
* Хранит членство в группе в базе данных «JIRA»
* Аутентифицирует пользователей
* Предоставляет API, который позволяет управлять пользователями, группами и членством в группах.
* Позволяет JIRA подключаться к внешним системам для извлечения данных о пользователях и группах.
* Сохраняет копию любых внешних данных в локальной БД для быстрого извлечения и синхронизации в фоновом режиме.

Как упоминалось ранее, «Seraph» делегирует «Embedded Crowd» для аутентификации пользователя (Проверяет, был ли введен правильный пароль, когда пользователь пытается войти в систему).

**PropertySet**

«OpenSymphony's PropertySet» - это фреймворк, который может хранить набор свойств (пары «ключ-значение») для конкретного «объекта» с уникальным идентификатором. «Сущность» может быть, чем угодно. Например, «UserPropertyManager» «JIRA» использует «PropertySet» для хранения предпочтений пользователя. Таким образом, в этом случае «сущность» является пользователем[17].

Каждое свойство имеет ключ (который всегда является java.lang.String) и значение, которое может быть:

* java.lang.String.
* java.lang.Long.
* java.util.Date.
* java.lang.Double.

Каждое свойство всегда связано с одним объектом. Что касается «PropertySet», то «сущность» имеет имя объекта и числовой идентификатор. Пока одна и та же комбинация имя и идентификатор используется для хранения значения и получения значения, все будет работать. В «JIRA PropertySet» используются следующие таблицы базы данных:

* «Propertyentry» - записывает имя объекта и идентификатор свойства, ключ и тип данных значения свойства. Каждая запись в этой таблице также имеет уникальный идентификатор.
* «Propertystring» - хранит значения типа String.
* «Propertydecimal» - хранит значения типа Double.
* «Propertydate» - хранит значения типа Date.
* «Propertynumber» - хранит значения типа Long.

**JIRA Utility and Manager Classes**

Почти вся бизнес-логика в «JIRA» реализована в java классах. Классы могут быть простыми служебными классами или объектами-диспетчерами.

Объекты диспетчеры в «JIRA» обычно имеют одну конкретную цель. Например, com.atlassian.jira.project.version.VersionManager используется для работы с версиями проекта, то есть создания, обновления, удаления и извлечения версий. Объекты диспетчеры используют множество внешних зависимостей, большинство из которых являются открытыми, некоторые из них разрабатываются «Atlassian» и, как правило, совместно используются между продуктами «Atlassian».

Начиная с «JIRA» 3.7 классы диспетчеры, как правило, обертываются соответствующим классом сервиса. Идея состоит в том, что любая проверка необходимой бизнес-логики осуществляется классами обслуживания, тогда как классы-менеджеры ответственны за фактическое выполнение действия[17]. На Рис. 15 показана схема архитектуры системы управления проектами «JIRA».

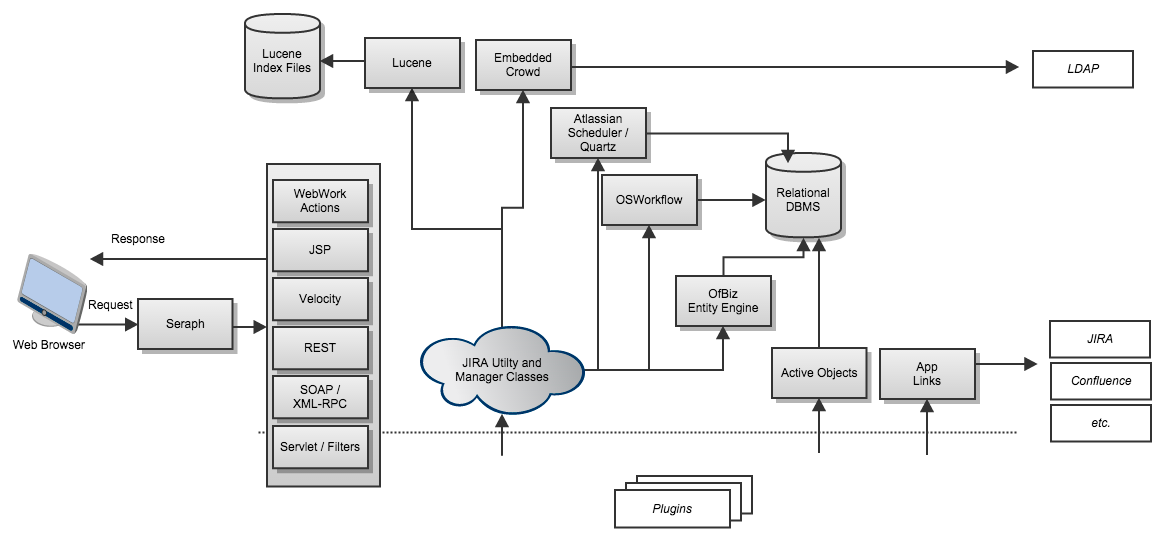


Рис. 15 Архитектура системы управления проектами «JIRA»

### Обзор литературы

При написании данной работы были использованы научная и учебно-методическая литература, статьи и материалы конференций, а также различные электронные ресурсы. Далее будет приведен разбор полученной информации из литературных источников об используемых технологиях при решении проблемы данной работы.

**Java**

В ходе реализации практической части данной работы необходимо было написать класс-реализатор, при его проектировании были применены знания о классах, методах, структуре и типах данных, полученные из руководства по Java 8 Герберта Шилдта.[18]

**Atlassian Jira**

Вся необходимая информация для решения поставленных задач об «Atlassian JIRA», а конкретно о структуре, архитектуре системы, абстракциях, подсистемах, рабочих процессах и многом другом, были получены из книги «Practical Jira Plugins» Matthew B. Doar.[19], а так же различных электронных источников, таких как: википедия[15], а так же официальные сайты, как предназначенные для пользователей[13][14][16], так и для разработчиков[17].

**Velocity**

Для отображения результатов работы плагина системы управления проектами «Atlassian JIRA» служат «Velocity» шаблоны, вся необходимая информация об этой технологии была получена из электронного источника, официального сайта проекта «Apache Velocity».[20]

**JavaScript**

Для отображения графиков, отображающих данные необходимые для принятия решений, был использован «JavaScript», вся необходимая информация об этом языке программирования была получена из книги «JavaScript: The Definitive Guide» Дэвида Флэнагана[21], а так же электронного источника, посвященного сбору интересных решений для красивого отображения информации с помощью «JavaScript» [22].

**Atlassian SDK**

Для разработки любого плагина для системы управления проектами «Atlassian JIRA» необходим «Atlassian SDK». «Atlassian SDK» — это набор инструментов разработки не только для «JIRA», но и для всей линейки продуктов «Atlassian» («Confluence», «Crowd» и т.д.) Так же, в последние годы выделяется общий для всех этих продуктов API — «Shared Access Layer». Несмотря на то что в разных «Atlassian-продуктах» используются разные библиотеки, разные подходы и т.д., в «Atlassian» понимают необходимость унификации, чтобы была возможность создавать общие дополнения.

«Atlassian SDK» представляет из себя «Apache Maven» с некоторыми дополнениями. Вместе с этим SDK идёт репозиторий с базовыми зависимостями «JIRA», файл «settings.xml» — конфигурация «Maven» для того, чтобы он брал зависимости из репозитория, а не скачивал их с «Maven Central». Также в этом файле прописан репозиторий «Atlassian Maven Repository», в котором лежат публичные библиотеки «Atlassian». Вся необходимая информация об «Atlassian SDK» была получена с официального сайта с документацией по «Atlassian SDK»[23], а так же из электронного источника[24].

**Apache Tomcat**

Входе изучения методов взаимодействия «Atlassian JIRA» с другими источниками данных потребовалось получить информацию об устройстве «Apache Tomcat» и «JNDI», вся необходимая информация была получена из официальной документации[25].

### Подготовка инструментов

Для выполнения цели данной работы потребовалось некоторое программное обеспечение и данные, на которых будет проводится тестирование работоспособности плагина. Ниже будет приведена инструкция, при выполнении которой будет проведена подготовка системы к разработке и тестированию плагина «Принятия решений» для системы управления проектами «JIRA».

**Java, установка**

Прежде всего необходимо установить среду выполнения «Java», для этого необходимо перейти на официальный сайт[26] и выбрать 1 из вариантов установки, а затем следовать инструкциям. После установки нужно задать переменную «JAVA\_HOME», отвечающую за путь до каталога, в которой была установлена «Java».

**Atlassian SDK**

После установки среды выполнения необходимо установить инструментарий для разработки плагинов для системы управления проектами «JIRA» «Atlassian SDK».

**Установка**

Сначала необходимо скачать последнюю версию SDK с официального сайта, а затем поэтапно выполнить инструкцию инсталлятора. Подробнее об установке «Atlassian SDK» можно узнать на официальном сайте для разработчиков[27].

**Основные команды**

«Atlassian SDK» представляет из себя модифицированный «Apache Maven», который является в свою очередь инструментом управления программным обеспечением, основанным на концепции объектной модели проекта или project object model(POM). Maven может управлять сборкой, компоновкой, отчетностью и документацией проекта. Управление осуществляется с помощью консольных команд, далее будут приведены команды, которые были использованы при реализации данной работы.

* Atlas-run, команда при выполнении которой будет проведена сборка, компиляция и запуск «Atlassian Jira» с включённым плагином. При первой запуске команды из репозитория будет выкачена версия «Atlassian Jira» с временной лицензией разработчика на 3 дня, которая будет обновляться при каждом выполнении команды.
* Atlas-package, команда при которой будет выполнена сборка и компиляция плагина, а так же при включенной функции QuickReload на сервере «Jira» будет проведено обновление плагина на сервере в автоматическом режиме.
* Atlas-clean, команда очищающая папку target в директории плагина.
* Atlas-create-jira-plugin, команда отвечающая за создание плагина, принимает на вход 4 параметра: groupId – ид компании, artifactId – ид плагина, version – версия плагина, package – путь до java класса реализатора плагина.
* Atlas-create-jira-plugin-module, команда отвечающая за создание модулей, которые являются основным каркасом для реализации функциональности плагина, принимает на вход id модуля, всего существует 38 стандартных модулей, каждый из которых имеет свой набор характеристик. Подробнее об этом написано в официальной документации. [28]

**NetBeans IDE, установка**

Следующим шагом будет установка IDE для эффективной работы с проектом, в данном случае это будет «NetBeans», так как в нем достаточно удобно реализована работа с java классами и зависимостями pom-файл. Для установки «NetBeans» необходимо перейти на официальный сайт разработчика[29], скачать актуальную версию программного обеспечения и установить, следуя инструкции.

**Notepad++, установка**

Для работы с «velocity» шаблонами и «JavaScript» скриптами был выбран редактор «Notepad++», так как он содержит все необходимое и поддерживает функцию авто закрытия html-тэгов, что упрощает работу с vm-файлами. Для его установки необходимо перейти на официальный сайт разработчика[30], выкачать последнюю версию программы и установить, следуя инструкциям.

**MySQL database установка**

В ходе работы было проведено так же изучение и тестирование технологии плагинов «JIRA» на предмет работы с внешними источниками данных, в роли такого источника данных был выбрана база данных «MySQL». Чтобы установить сервер «MySQL database» необходимо перейти на официальный сайт[31], скачать стабильную версию установщика, а затем установить, следуя инструкциям.

**MySQL workbench**

**Установка**

«MySQL database» после установки позволяет редактировать данные только с помощью консоли, это не слишком удобно, поэтому в целях увеличения эффективности работы с базой данных было решено установить «MySQL workbench», данное программное обеспечение представляет из себя инструмент для проектирования баз данных. Для установки «MySQL workbench» необходимо перейти на официальный сайт разработчика[32], выкачать последнюю версию установщика, а затем установить, следуя инструкции.

**Конфигурирование БД**

Для проведения тестирования технологии работы плагинов «JIRA» с внешними источниками данных была создана схема «jiradb», затем была создана таблица «issue\_additional\_fields» с полями «issue\_id» – уникальный идентификатор, по нему проходит сравнение с полем «issue key» задачи из базы данных «JIRA», «day\_to\_end» – поле, отвечающее за количество дней до завершения задачи, и поле «priority» – отвечающее за приоритет задачи. На Рис. 16 показана структура таблицы и данные, которые в ней хранятся.

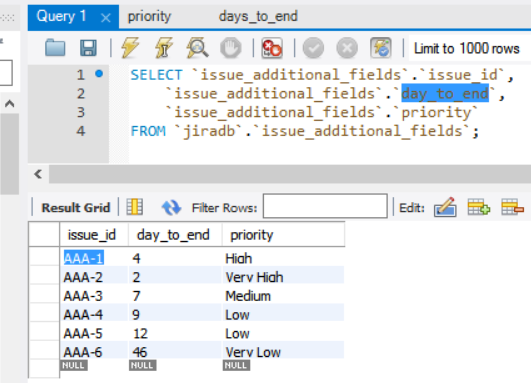


Рис. 16 Таблица «issue\_additional\_fields»

**«Atlassian Jira»**

**Установка**

Следующим этапом будет установка непосредственно самой системы «Atlassian Jira», для этого необходимо перейти на официальный сайт[17], оформить пробную лицензию, сгенерировать ключ на основе данной лицензии, затем скачать архив с последней версией программного обеспечения и установить его, следуя инструкциям.

После установки «Atlassian Jira» необходимо перейти в каталог, в который система была установлена и запустить файл run\_jira.bat, данное действие запустит развертывание «JIRA» и последующий запуск локального сервера.

После запуска сервера необходимо зайти на сервер с помощью веб браузера и ввести логин и пароль, которые были выбраны при установке, и проверить все ли функции программного обеспечения работают корректно, в случае возникновении проблем необходимо обратиться на официальный сайт[17].

**Плагин NFEED**

После установки и базовых проверок корректности работы сервера «Atlassian Jira» необходимо установить плагин для работы с внешними данными «NFEED», данный плагин – один из вариантов решения проблемы получения информации из внешних источников данных, сама проблема будет подробно рассмотрена во 2 главе данной работы. Для того чтобы установить данный плагин необходимо скачать его с официального сайта разработчика[33], а так же получить на него пробную лицензию на 30 дней. После того, как плагин был скачан, его необходимо распаковать в любую директорию, затем необходимо зайти в настройки сервера «Atlassian Jira», перейти на вкладку «плагины», затем нажать кнопку установить новый плагин и выбрать путь до директории, в которую был распакован плагин. После этого необходимо выполнить настройку 2 пользовательских полей, в которые будут выбраны данные из «MySQL database». Конфигурация полей происходит из меню «пользовательские поля», в данном меню необходимо создать пользовательское поле, затем выбрать тип данного поля, в данном случае тип поля будет «NFEED standard field», затем нужно выбрать область видимости поля, в данном случае необходимо выбрать пункт issue. После создания таким образом 2 полей: «priority» и «days\_to\_end» их необходимо наполнить данными из внешнего источника данных, для этого необходимо перейти в меню настройки плагина «NFEED» и выбрать вкладку настройка пользовательских полей. На данной вкладке с помощью контекстного меню настраивается путь до внешнего источника данных, а затем предлагается ввести sql-запрос для выборки необходимых данных в поле, либо можно так же воспользоваться контекстным меню и с помощью него составить из блоков необходимый запрос. Подробнее о работе плагина «NFEED» можно узнать на официальном сайте с документацией[34]. На Рис. 17 показано меню конфигурации внешнего источника данных, на Рис. 18 показано контекстное меню выбора таблицы из внешнего источника данных, на Рис. 19 показано меню тестирования sql-запроса.

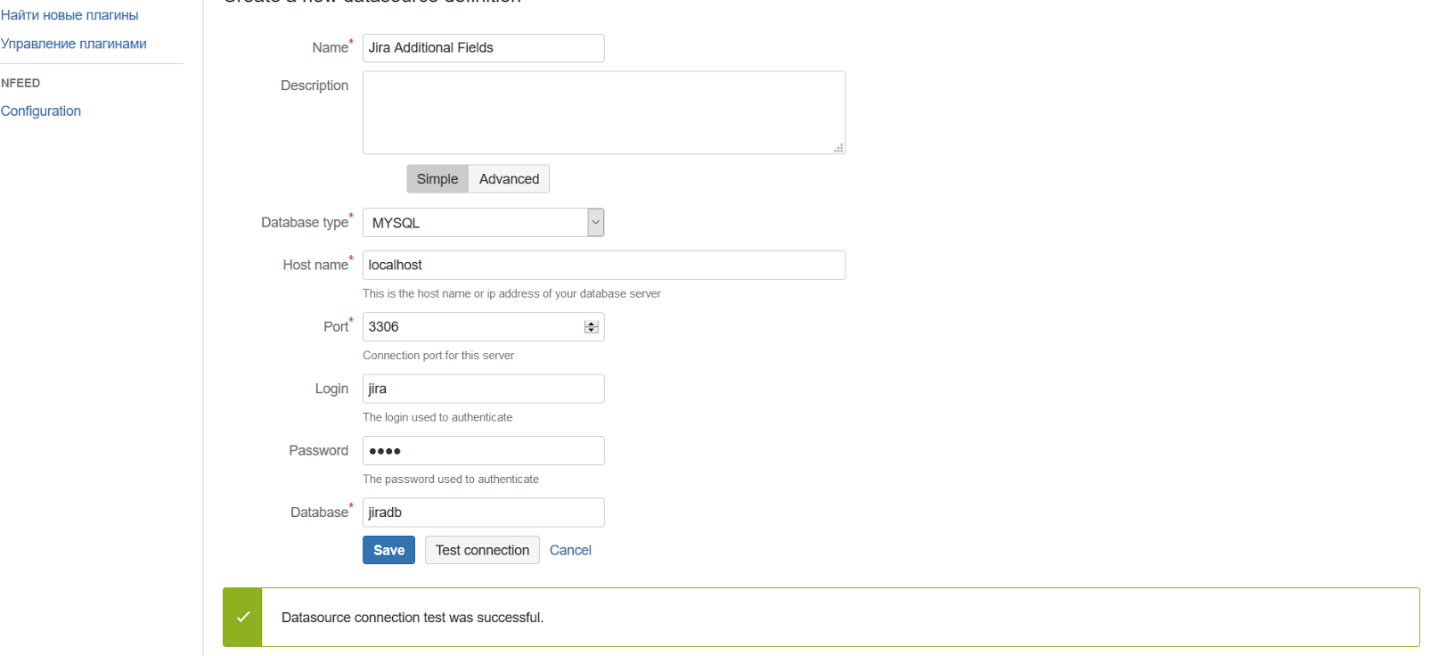


Рис. 17 Меню конфигурации внешнего источника данных

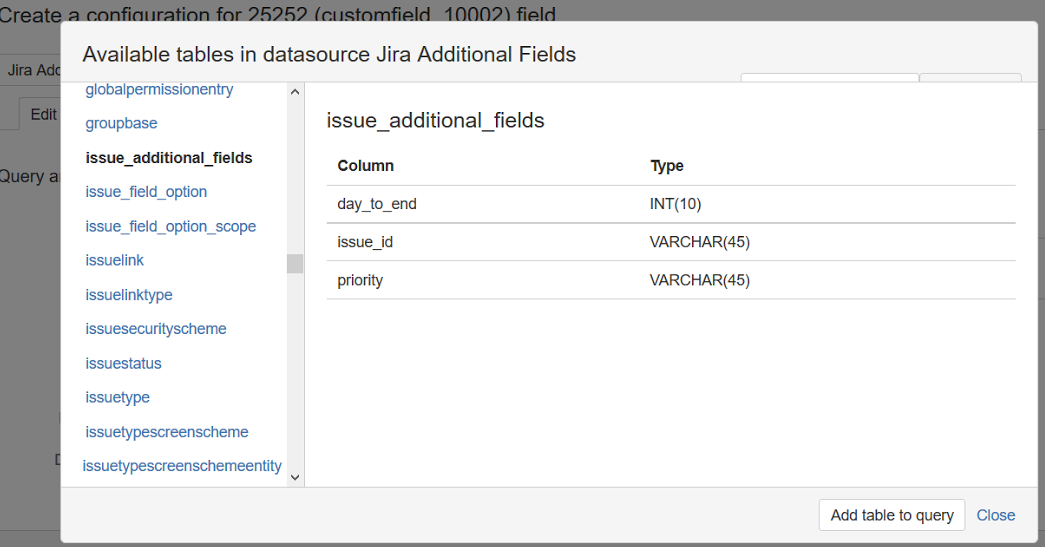


Рис. 18 Меню выбора таблицы из внешнего источника данных

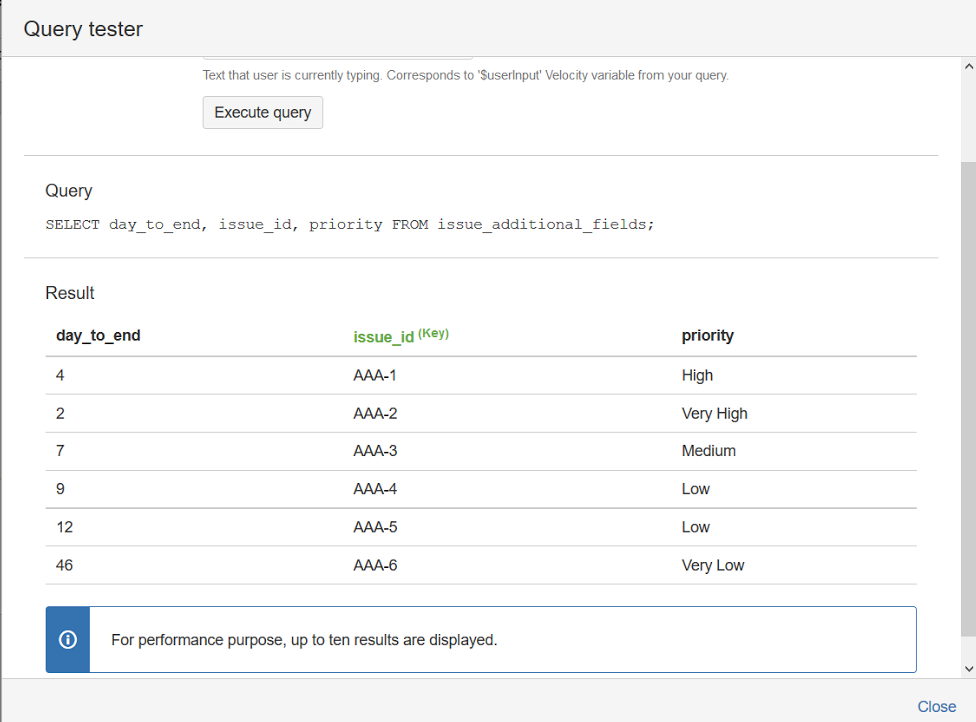


Рис. 19 Меню тестирования sql-запроса данных из таблицы

**Конфигурирование проектов и задач**

Для того чтобы работать с «Atlassian Jira» необходимо произвести конфигурацию проектов и задач, далее будет описан процесс конфигурации. Создание проекта происходит из вкладки проекты, необходимо нажать на кнопку «создать проект», выбрать 1 из 3 вариантов проектов, в данном случае наиболее подходящий тип «управление задачами», подробнее о типах проектов можно узнать из документации[13], после выбора типа проекта необходимо ввести название проекта и «project key», о котором было написано ранее в данной работе. На Рис. 20 продемонстрировано меню создания проекта.

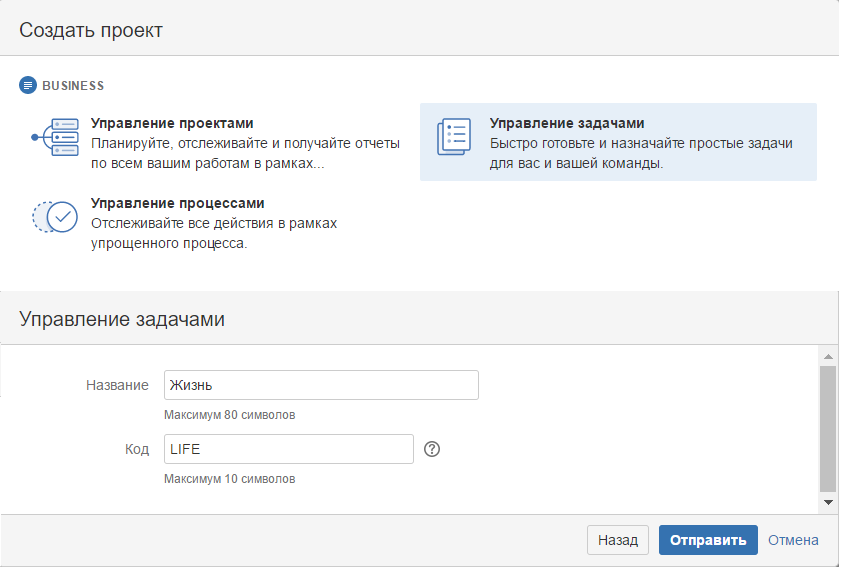


Рис. 20 Меню создания проекта

После создания проекта необходимо создать задачи для данного проекта, для этого необходимо нажать на кнопку «создать», на Рис. 21 показано расположение данной кнопки в пустом проекте.

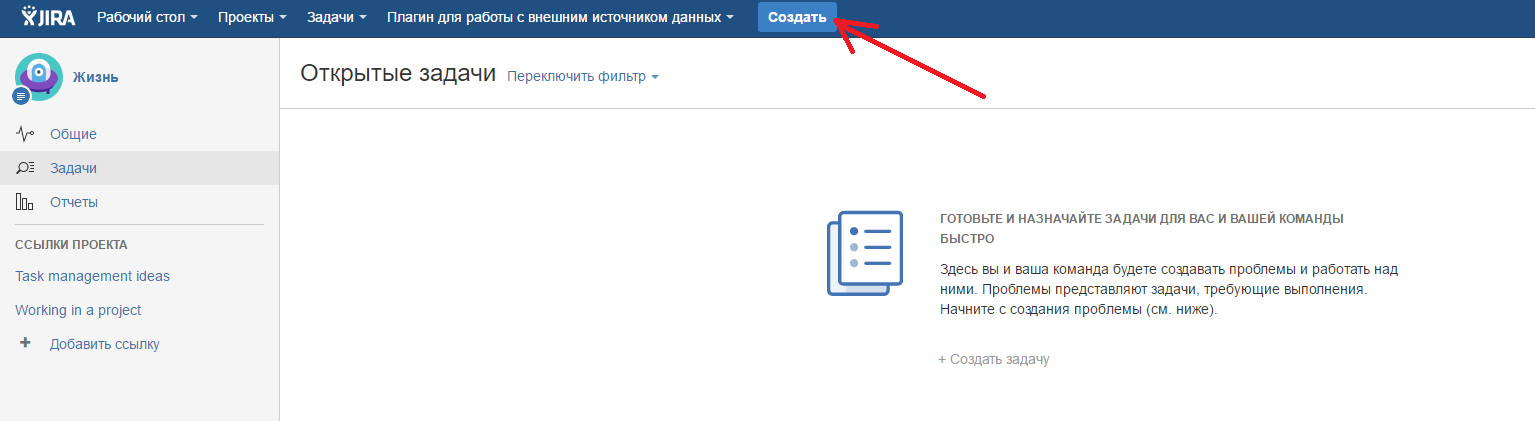
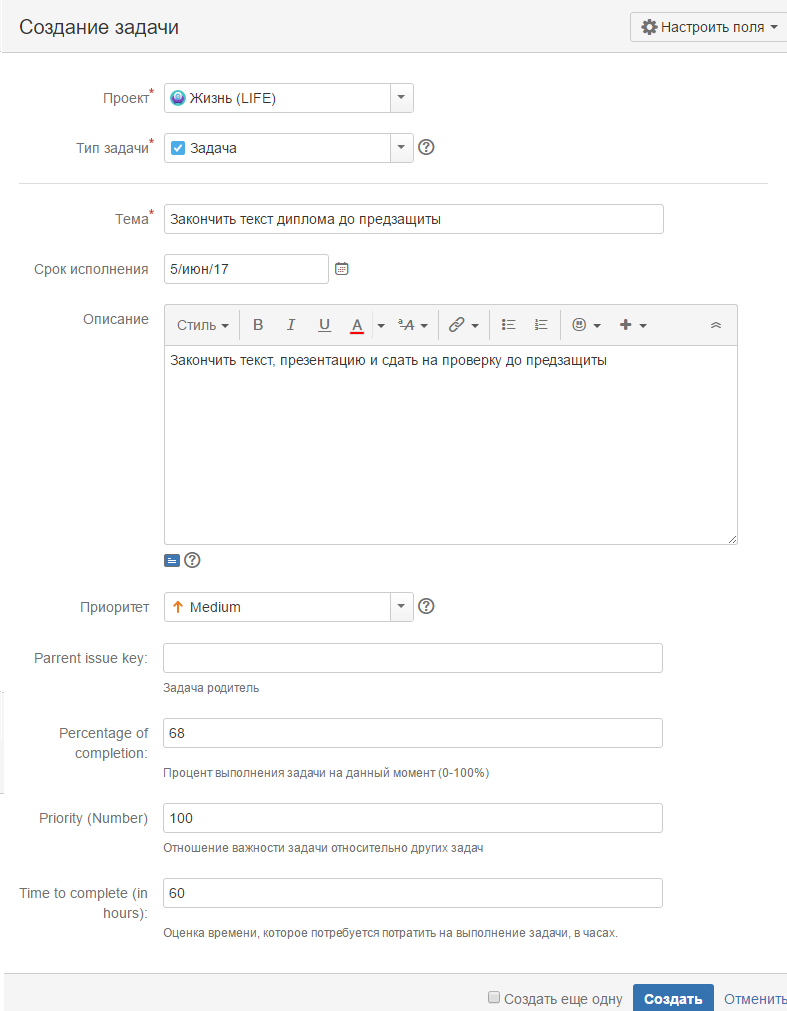


Рис. 21 Пустой проект, расположение кнопки создания задачи

При нажатии на данную кнопку открывается меню создания задачи, в данном меню необходимо заполнить все необходимые поля и сохранить задачу, на Рис. 22 продемонстрировано меню создания задачи.

Рис. 22 Меню создание задачи

После сохранения данных задача будет создана и отобразиться в главном меню, на Рис. 23 продемонстрирована созданная задача с Рис. 22.

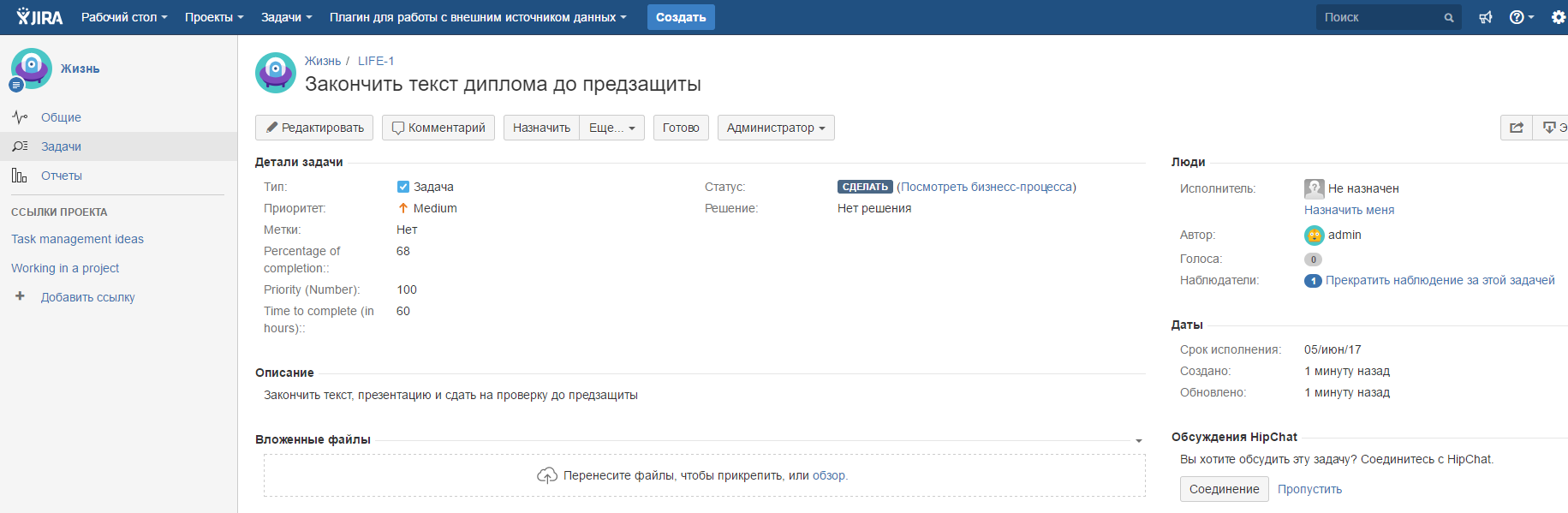


Рис. 23 Созданная задача

**Заполнение данными**

Всего для данной работы было создано 2 проекта, первый из них это проект «AAA» - тестовый проект для проверки различных элементов технологии плагинов системы управления проектами «Atlassian Jira», данные из него будут фигурировать только в части работы по изучению взаимодействия плагинов с внешними источниками данных, на Рис. 24 показаны все задачи проекта со всеми используемыми данными.

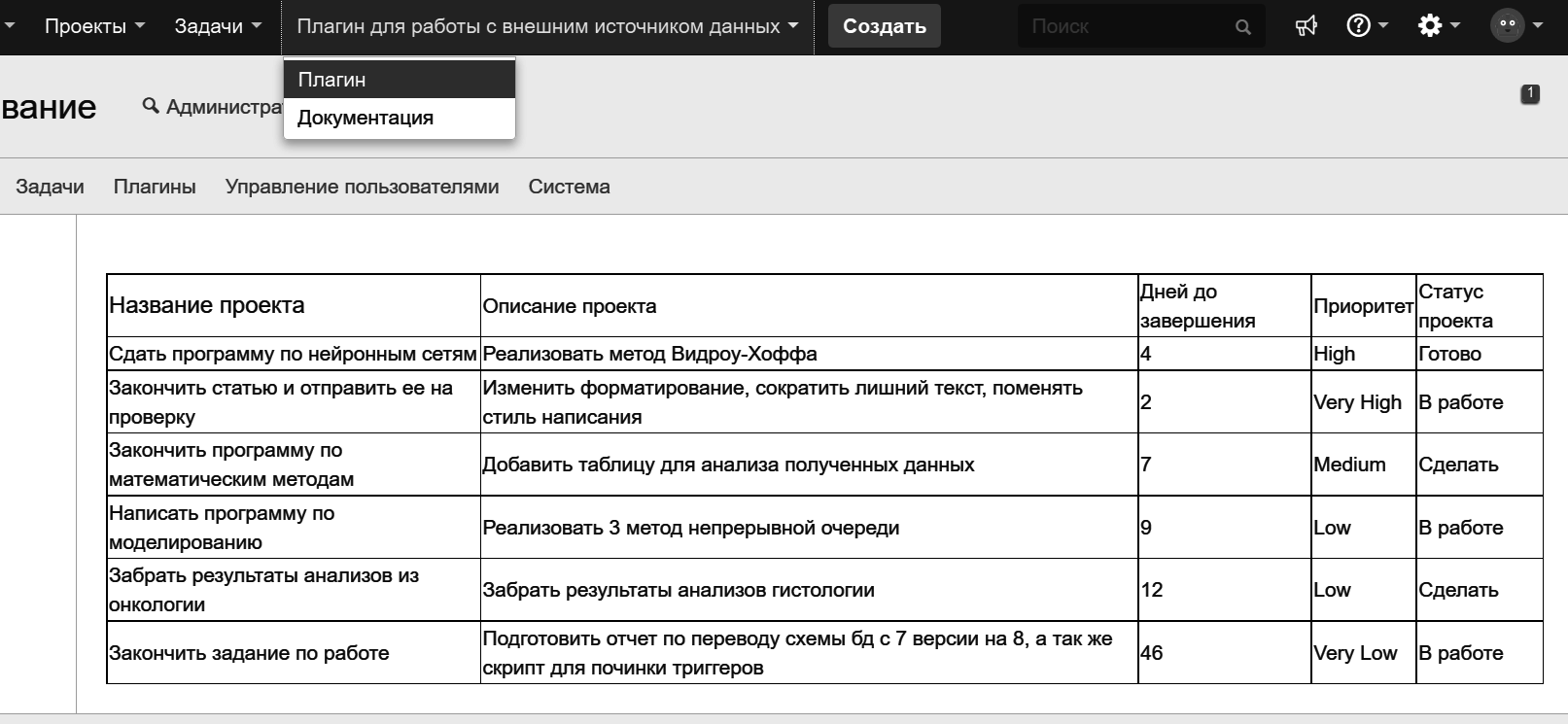


Рис. 24 Проект «AAA»

Вторым проектом стал «Жизнь», он содержит все задачи, которые нужно решить автору данной работы до июля 2017 года, на Рис. 25 продемонстрирован список этих задач.

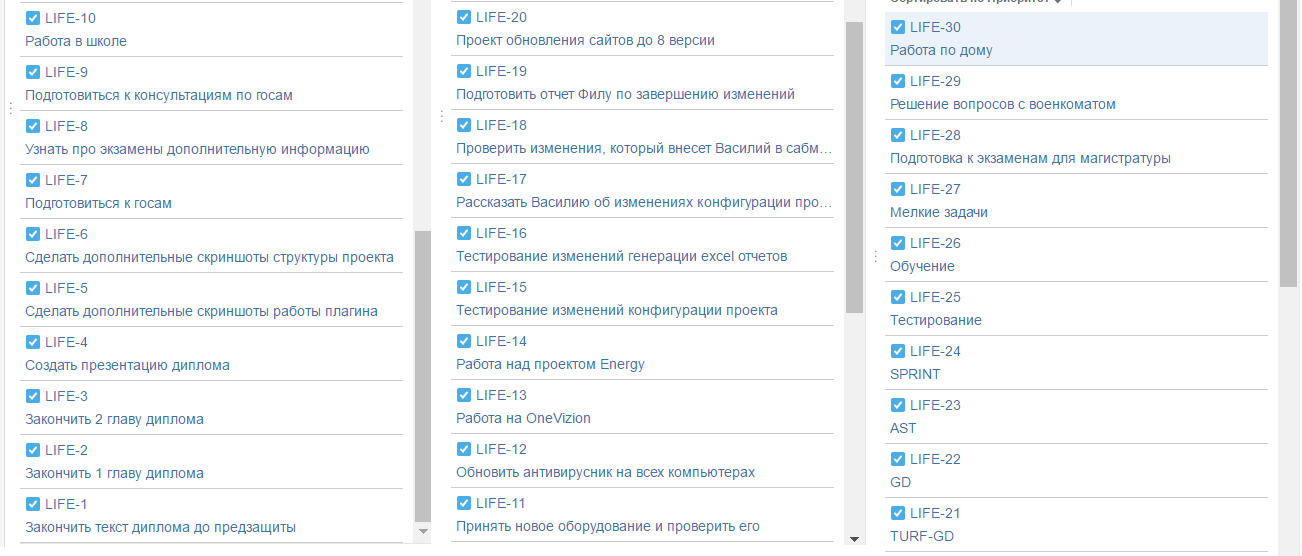


Рис. 25 Проект «Жизнь»

# 2. Реализация

## 2.1. Требования к приложению

К плагину «Принятие решений» были предъявлены следующие требования:

• Возможность отображать все задачи проекта в виде иерархической структуры(дерева);

• Возможность быстро определить приоритет задачи относительно других задач на основе критерия «Принятие решений»;

• Возможность получить подробное описание задачи в случае необходимости;

• Возможность построения сравнительных графиков по иерархическим уровням вложенности подзадач.

## 2.2. Архитектура

Перед тем, как реализовывать плагин необходимо определиться с его архитектурой, а так же структурой всех его составных частей, далее будет представлено описание этого этапа работы.

### 2.2.1. Описание структуры плагина

Любой плагин системы управления проектами «Atlassian Jira» состоит из 3 основных частей, это файл-дескриптор, отвечающий за взаимодействие между всеми элементами плагина, а так же описывающий стандартные элементы вроде кнопок, источников данных, связей, классов реализаторов и многое другое, подробный разбор файла-дескриптора будет проведен далее, следующая важная часть плагина, это java класс реализатор, именно в нем хранятся все необходимые данные и происходят все расчеты, так же в нем реализован алгоритм «Принятия решений», и последней значительной частью плагина является velocity шаблон, в котором описывается весь внешний вид плагина.

### 2.2.2. Описание структуры файла-дескриптора

Все плагины для системы управления проектами «Atlassian Jira» обязаны включать в себя файл-дескриптор «atlassian-plugin.xml», далее будет описан процесс загрузки jar-файла плагина на сервер:

* Происходит загрузка кода плагина из меню добавления нового плагинов на сервере;
* Код загружается и попадает в OSGI-контейнер;
* Плагин начинает работать, для того, чтобы понять, что делает плагин, «JIRA» использует файл-дескриптор.

В случае отсутствии файла-дескриптора будет выведена соответствующая ошибка при попытке развернуть плагин на сервере «Atlassian Jira». Таким образом любой плагин не может обойтись без файла-дескриптора. Рассмотрим подробнее файл «atlassian-plugin.xml» плагина «Принятие решений», который был автоматически сгенерирован консольной командой «Atlassian SDK», а затем расширен в ручную, в Листинге 1 приведен раздел информации по проекту, а конкретно дано описание проекта, версия, имя продавца, эти данные берутся из pom-файла проекта, а так же здесь описываются иконка и лого.

|  |
| --- |
| <plugin-info>  <description>${project.description}</description>  <version>${project.version}</version>  <vendor name="${project.organization.name}" url="${project.organization.url}"/>  <param name="plugin-icon">images/pluginIcon.png</param>  <param name="plugin-logo">images/pluginLogo.png</param>  </plugin-info> |

Листинг 1 «atlassian-plugin.xml», информация по проекту

В Листинге 2 приведено описание модуля «web-work», который отвечает за вывод результата работы плагина, в зависимости от результата произведенных вычислений внутри класса реализатора плагина будет использован 1 из 3 velocity шаблонов.

|  |
| --- |
| <webwork1 key="my-webwork-module" name="My Webwork Module" i18n-name-key="my-webwork-module.name">  <description key="my-webwork-module.description">The My Webwork Module Plugin</description>  <actions>  <action name="com.atlassian.externaldbplugin.jira.webwork.MyWebworkModuleAction" alias="MyWebworkModuleAction">  <view name="success">/templates/my-webwork-module/success.vm</view>  <view name="input">/templates/my-webwork-module/input.vm</view>  <view name="error">/templates/my-webwork-module/error.vm</view>  </action>  </actions>  </webwork1> |

Листинг 2 «atlassian-plugin.xml», модуль webwork

В Листинге 3 приведено описание всех элементов управления плагином, которые представляют из себя 1 панель с выпадающими 2 кнопками, одна из которых перенаправляет пользователя на сайт с документацией, а вторая вызывает класс реализатор плагина «MyWebworkModuleAction.jspa», который будет рассмотрен далее.

|  |
| --- |
| <web-section key="my\_links\_section" name="My Links Main Section" location="my\_links\_link" weight="10"/>  <web-item key="my\_links\_link" name="Link on My Links Main Section" section="system.top.navigation.bar" weight="47">  <label>Плагин Принятие решений</label>  <link linkId="my\_links\_link"></link>  </web-item>  <web-item key="My\_Action\_Action" name="MyAction" section="my\_links\_link/my\_links\_section" weight="10">  <label>Плагин</label>  <link linkId="My\_Action\_Action">/secure/MyWebworkModuleAction.jspa</link>  </web-item>  <web-item key="documentation\_link" name="Documentation Web Site" section="my\_links\_link/my\_links\_section" weight="10">  <label>Документация</label>  <link linkId="documentation\_link">https://developer.atlassian.com/docs/</link>  </web-item> |

Листинг 3 «atlassian-plugin.xml», описание элементов управления плагина

### 2.2.3. Описание структуры java класса реализатора

Вся бизнес логика плагина системы управления проектами «Atlassian Jira» реализуется в java классах, в данном проекте такой класс всего 1, и это «MyWebworkModuleAction.jspa», подробнее о нем будет написано в последующих пунктах, а пока что будет рассмотрена 1 ключевая особенность взаимодействия java класса с velocity шаблонами в «JIRA», чтобы вывести на экран любую переменную из класса реализатора необходимо выполнить 2 условия:

* Переменная обязательно должна быть private
* Для данной переменной должны быть объявлены get и set методы

В Листинге 4 приведен пример описание 1 из переменных, которые выводятся на экран при удачном завершении работа плагина «Принятия решений».

|  |
| --- |
| private int maxlevel;  public void setMaxlevel(int maxlevel){this.maxlevel = maxlevel;}  public int getMaxlevel(){return this.maxlevel;} |

Листинг 4 «MyWebworkModuleAction.jspa», описание переменной «maxlevel»

### 2.2.4. Описание структуры velocity шаблона

«Velocity» шаблоны в системе управления проектами «Atlassian Jira» отвечают за вывод информации полученной из java класса реализатора, для того чтобы вывести переменную на экран:

1. Нужно ее описать соответствующим способом в «java» классе (Пример описания рассмотрен в предыдущем пункте)
2. Обратиться к ней соответствующим способом из «velocity» шаблона (Пример подобного обращения в Листинге 5)

|  |
| --- |
| Максимальный уровень вложенности подзадачи: $maxlevel <br>  #foreach($issue in $issues)  #if ($issue.getIssueParentkey() == "No parent")  ['$issue.getIssueKey()', $issue.getIssueContribution()]  #end |

Листинг 5 «success.vm», вывод переменной «maxlevel» и полей класса «issue», у которых нет родителей

Подробнее о выводе информации в «velocity» шаблоне можно узнать из документации на официальном сайте[20].

## 2.3. Плагины в «Atlassian JIRA»

Исходя из определения, плагин – это независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования ее возможностей[35]. В системе управления проектами «Atlassian Jira» плагин так же представляет из себя независимую от основного сервера программный модуль, который выполняет некоторые действия. Подключение плагинов к основной системе, как было описано ранее, происходит из меню «плагины», для обновления кода плагина есть 2 варианта:

* Вручную перезагрузить новый jar-файл плагина из меню «плагины»;
* Настроить функцию «QuickReload» на сервере «Jira», в данном случае сервер будет автоматически обновлять и разворачивать код плагина при изменении jar-файла плагина.

## 2.4. Технология доступа к внешним источникам данных

Одной из целей данной работы было изучение и тестирование технологии взаимодействия плагинов системы «Atlassian Jira» с внешними источниками данных. В ходе изысканий было выявлено 3 основных способа доступа к внешним данным из плагина, это:

* Доступ с помощью «JDBC» драйвера
* Доступ через «JNDI» соединение
* Доступ с помощью стороннего плагина «NFEED»

Далее будет приведен обзор каждого из этих способов.

**JDBC драйвер**

Простейшим способом получить данные из внешнего источника данных в плагине «JIRA» является использование «JDBC» драйвера, для использования этого способа необходимо совершить ряд действий:

* Создать пользователя базы данных, которая будет использована, как внешний источник данных.
* Создать и сконфигурировать базу данных, которая будет использоваться, как внешний источник данных.
* Скопировать подходящий «JDBC» драйвер в директорию lib на сервере «Atlassian Jira».
* Добавить в файл dbconfig.xml сервера дополнительные строки, описывающие новый источник данных.

После данных действий можно получить данные из внешнего источника данных внутри плагина стандартным способом. Подробнее о настройке «JDBC» соединения можно узнать на официальном сайте[36].

**JNDI соединение**

«Java Naming and Directory Interface» (JNDI) – это набор «Java API», который позволяет «Java» клиентам открывать и просматривать данные и объекты по их именам. Чтобы использовать данное «API» в системе управления проектами «Atlassian Jira» необходимо описать данное соединение в 2 файлах: server.xml и dbconfig.xml, после данных действий соединение будет настроено и данные можно будет получить из внешнего источника внутри плагина. Подробнее о настройке «JNDI» соединения можно узнать на официальном сайте[37].

**Плагин NFEED**

Последним из рассматриваемых способов получения данных из внешнего источника стал сторонний плагин «NFEED», данный плагин позволяет получать данные, как с помощью «JNDI» соединения, так и с помощью «JDBC» драйвера, но в отличие от предыдущих способов, здесь не нужно в ручную редактировать файлы на сервере. Кроме того, плагин так же добавляет несколько полезных функций, таких как тестирование соединения, тестирование запросов, а так же набор типов пользовательских полей, которые позволяют получать данные из внешних источников данных без дополнительного кода внутри плагина. Подробнее о плагине «NFEED» можно узнать на официальном сайте плагина[34].

## 2.5. Реализация плагина «Принятие решений»

В данном параграфе будет подробно описываться непосредственно сам процесс реализации плагина «Принятие решений».

### 2.5.1. Создание каркаса плагина

Создание любого плагина для системы управления проектами «Atlassian Jira» начинается с создания каркаса плагина, каркас плагина создается командой «atlas-create-jira-plugin», затем необходимо ввести «groupId», «artifactId», «version» и «package», подробнее об этим переменных было написано в параграфе по основным командам «Atlassian SDK». На Рис. 26 изображен процесс создания каркаса плагина.

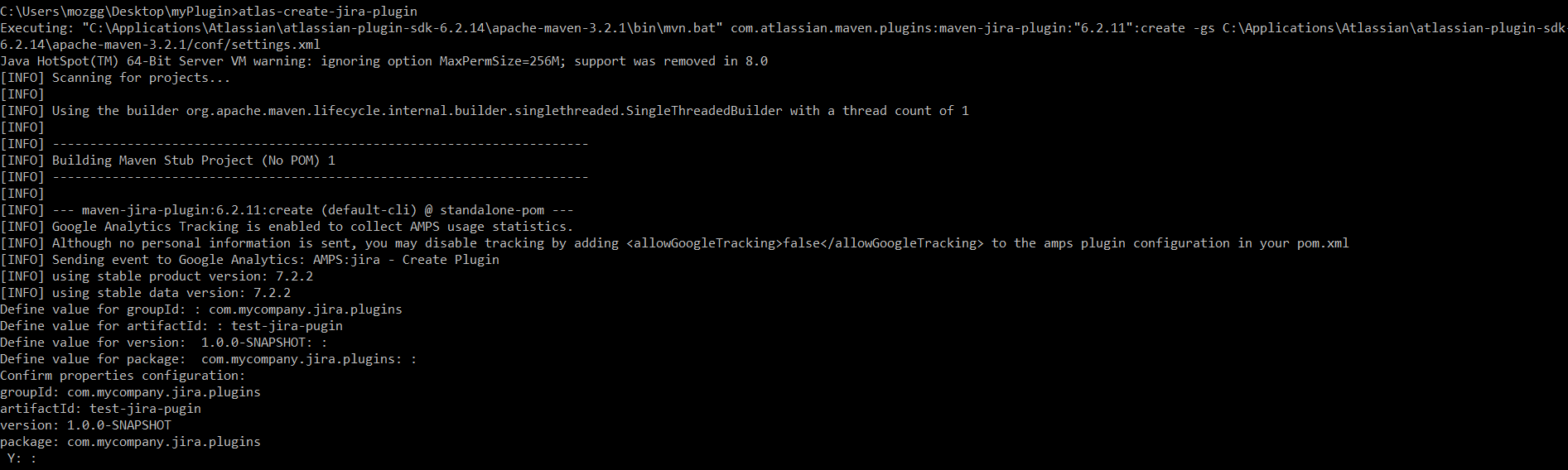


Рис. 26 Создание каркаса плагина

### 2.5.2. Расширение каркаса модулями

После создания каркаса плагина его необходимо расширить дополнительными модулями, расширение каркаса происходит с помощью команды «atlas-create-jira-plugin-module», затем необходимо ввести номер типа модуля, в данном случае это 25 «Web Item», модуль отвечающий за элемент кнопку, затем необходимо ввести имя модуля, имя секции в которой будет расположена кнопка, в данном случае это «operations-top-level», верхняя панель меню, и ввести путь до java класса реализатора, который будет вызван при нажатии на данную кнопку. На Рис. 27 показан процесс создания модуля плагина.

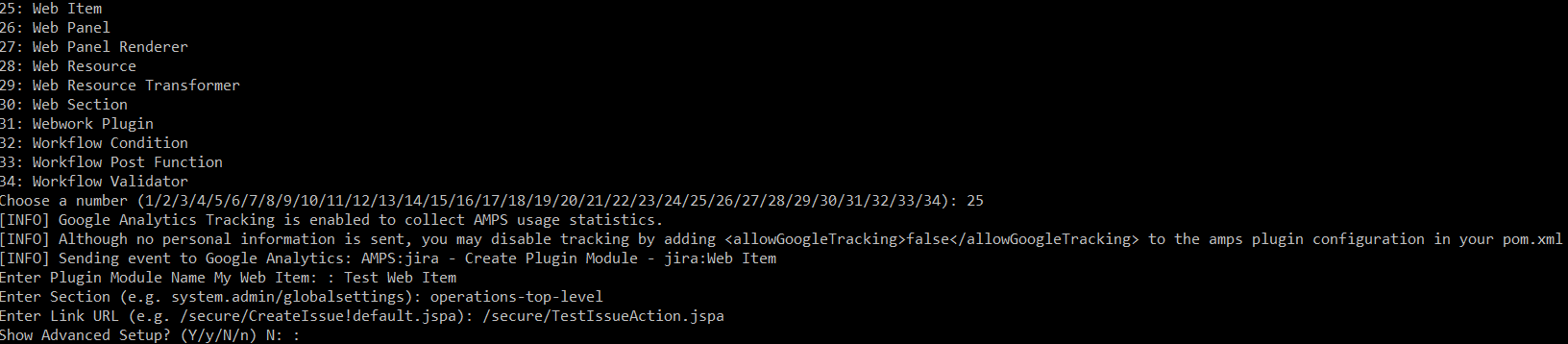


Рис. 27 Создание модуля плагина

### 2.5.3. Редактирование файла дескриптора

После создания каркаса и всех необходимых модулей плагина, необходимо отредактировать файл дескриптор «atlassian-plugin.xml», описание стуктуры данного файла приводилось ранее в параграфе «архитектура», далее в Листинге 6 будет приведен полный код файла «atlassian-plugin.xml» плагина «Принятие решений». В данном плагине необходимо было отредактировать только модуль Web Work, были добавлены дополнительные кнопки, а так же были отредактированы пути до velocity шаблонов.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <atlassian-plugin key="${atlassian.plugin.key}" name="${project.name}" plugins-version="2">  <plugin-info>  <description>${project.description}</description>  <version>${project.version}</version>  <vendor name="${project.organization.name}" url="${project.organization.url}"/>  <param name="plugin-icon">images/pluginIcon.png</param>  <param name="plugin-logo">images/pluginLogo.png</param>  </plugin-info>  <!-- add our i18n resource -->  <resource type="i18n" name="i18n" location="myExternalDbPlugin"/>  <!-- add our web resources -->  <web-resource key="myExternalDbPlugin-resources" name="myExternalDbPlugin Web Resources">  <dependency>com.atlassian.auiplugin:ajs</dependency>  <resource type="download" name="myExternalDbPlugin.css" location="/css/myExternalDbPlugin.css"/>  <resource type="download" name="myExternalDbPlugin.js" location="/js/myExternalDbPlugin.js"/>  <resource type="download" name="images/" location="/images"/>  <context>myExternalDbPlugin</context>  </web-resource>  <webwork1 key="my-webwork-module" name="My Webwork Module" i18n-name-key="my-webwork-module.name">  <description key="my-webwork-module.description">The My Webwork Module Plugin</description>  <actions>  <action name="com.atlassian.externaldbplugin.jira.webwork.MyWebworkModuleAction" alias="MyWebworkModuleAction">  <view name="success">/templates/my-webwork-module/success.vm</view>  <view name="input">/templates/my-webwork-module/input.vm</view>  <view name="error">/templates/my-webwork-module/error.vm</view>  </action>  </actions>  </webwork1>    <web-section key="my\_links\_section" name="My Links Main Section" location="my\_links\_link" weight="10"/>  <web-item key="my\_links\_link" name="Link on My Links Main Section" section="system.top.navigation.bar" weight="47">  <label>Плагин Принятие решений</label>  <link linkId="my\_links\_link"></link>  </web-item>  <web-item key="My\_Action\_Action" name="MyAction" section="my\_links\_link/my\_links\_section" weight="10">  <label>Плагин</label>  <link linkId="My\_Action\_Action">/secure/MyWebworkModuleAction.jspa</link>  </web-item>  <web-item key="documentation\_link" name="Documentation Web Site" section="my\_links\_link/my\_links\_section" weight="10">  <label>Документация</label>  <link linkId="documentation\_link">https://developer.atlassian.com/docs/</link>  </web-item>    </atlassian-plugin> |

Листинг 6. Файл дескриптор плагина «Принятие решений»

### 2.5.4. Бизнес логика

Далее будет приведено описание бизнес логики плагина «Принятие решений», вся логика плагина делится на ряд функциональных блоков, отвечающих за свои процессы:

**Блок 1**, получение всех задач проекта с помощью «Jira java API», а так же считывание всех необходимых полей в класс «Issue», а конкретно полей:

* key – ключ задачи;
* parentkey – ключ задачи родителя, если у задачи нет родителя, то parentkey = null;
* level - уровень подзадачи, для задач у которых нет родителей значение level = 0, для остальных задач значение level рассчитывается из формулы level = parentlevel + 1;
* name – название задачи;
* desc – описание задачи;
* status – текущий статус задачи;
* priority – поле, отвечающее за приоритет задачи относительно других задач;
* urgency – поле, отвечающее за срочность задачи относительно других задач, рассчитывается на основе даты в которую данную задачу необходимо закончить;
* complexity – поле, отвечающее за сложность задачи относительно других задач, рассчитывается на основе времени, которое необходимо потратить на решение данной задачи;
* weight - поле, отвечающее за общий вклад задачи относительно других задач, является 1 из 2 полей, на основе которого рассчитывается критерий «Принятие решений», формула расчета данного поля будет приведена в соответствующем блоке;
* value – поле, отвечающее за текущий вклад задачи относительно других задач, является 1 из 2 полей, на основе которого рассчитывается критерий «Принятие решений», формула расчета данного поля будет приведена в соответствующем блоке;
* contribution – поле, представляющее из себя критерий «Принятие решений», формула расчета данного поля будет приведена в соответствующем блоке;
* time\_to\_complete – поле, показывающее количество часов, которые потребуются потратить на завершение текущей задачи;
* time\_to\_end – поле, показывающее количество часов, которые остались до завершения текущей задачи;
* percantage\_of\_completion – поле, показывающее текущий процент готовности текущей задачи;

**Блок 2**, нормирование полей, следующий логический блок отвечает за нормирование 3 полей: priority, urgency и complexity. Данные поля будут использованы после нормировки при расчете поля weight. Нормирование происходит по уровням вложенности подзадачи, начиная от 0 и заканчивая максимальным уровнем. Нормирование проходит по стандартной формуле, за 1 проход происходит суммирование всех значений полей текущего уровня в соответствующее поле, например sumpriority = сумма всех значений поля priority текущего уровня, за 2 проход значение поля priority заменяется по формуле priority = priority / sumpriority. Аналогичные действия происходят с полями urgency и complexity.

**Блок 3**, расчет критерия «Принятие решений», данный логический блок отвечает за расчет непосредственно самого критерия contribution, а так же 2 вспомогательных полей: weight и value. Все поля рассчитываются за 1 проход по всем задачам, поле weight рассчитывается по формуле weight = (urgency + priority + complexity) / 3 \* scale, поле scale задается для всех задач 0 уровня перед началом работы плагина и рассчитывается по формуле scale = parent weight, если уровень больше 0. Поле value рассчитывается по формуле value = weight \* percantage\_of\_completion / 100, а итоговый критерий рассчитывается по формуле contribution = weight – value.

В результате тестирования данного критерия было установлено, что он оптимален для всех случаев кроме пограничных, когда нормированные значения полей urgency, priority и complexity близки к 0 либо к 1. Поэтому был так же выработан 2 критерий, решающий данную проблему. Он рассчитывается аналогичным способом с предыдущим критерием до момента расчета поля weight в 3 логическом блоке. Новая формула для расчета критерия:

Поле value рассчитывается по формуле value = weight \* percantage\_of\_completion / 100, а итоговый критерий рассчитывается по формуле contribution = weight – value.

### 2.5.5. Написание java класса реализатора

Вся бизнес логика плагина реализуется в java классе реализаторе, в данном плагине реализован 1 класс issue, а так же набор методов в которых происходят считывание данных и все необходимые рассчеты. В Листинге 7 описаны все переменные класса issue.

|  |
| --- |
| private String key; //ключ задачи  private String parentkey; //ключ родителя задачи  private int level; //уровень вложенности подзадачи  private String name; //название задачи  private String desc; //описание задачи  private String status; //статус задачи  private double complexity; //сложность задачи  private double priority; //приоритет задачи(число)  private double urgency; //срочность задачи  private double weight; //общий вклад задачи  private double value; //текущий вклад задачи  private double contribution; // критерий принятия решений  private double time\_to\_complete; // время для завершения задачи  private double time\_to\_end; // время до конца задачи  private double percantage\_of\_completion; // процент выполнения задачи |

Листинг 7 Описание переменных класса issue

Для каждого из описанных переменных существует свой get и set метод, в Листинге 8 приведены примеры таких методов для переменной key. Для остальных переменных такие методы аналогичны.

|  |
| --- |
| public void setIssueKey(String key){this.key = key;}  public String getIssueKey(){return this.key;} |

Листинг 8 Описание get и set методов переменной key класса issue

Считывание данных в java классе происходит с помощью Jira java API, в Листинге 9 приведен пример доступа к данным из базы данных Jira.

|  |
| --- |
| ComponentAccessor componentAccessor = new ComponentAccessor();  ApplicationUser user = componentAccessor.getUserManager().getUserByName("admin");  IssueManager issueManager = componentAccessor.getIssueManager();  CustomFieldManager customFieldManager = componentAccessor.getCustomFieldManager();  MutableIssue issueObject;  Status status;  SimpleStatus simplestatus;  CustomField customFieldObject;  String projectname = "AAA-";  int id = 1;  int count = 0;  issueObject = issueManager.getIssueByCurrentKey(projectname + id); |

Листинг 9 Получение доступа к данным задачи AAA-1

После получение доступа к задачи необходимо считать данные из ее полей, в Листинге 10 приведен пример доступа к полям Issue Key и пользовательскому полю Time to complete, доступ к остальным полям задачи происходит аналогичным способом.

|  |
| --- |
| //Getting Issue Key  temp.setIssueKey(issueObject.getKey());  //Getting Time\_to\_complete  customFieldObject = customFieldManager.getCustomFieldObject("customfield\_10201");  temp.setIssueTime\_to\_complete( Double.parseDouble(issueObject.getCustomFieldValue(customFieldObject).toString())  ); |

Листинг 10 Получение данных полей Issue Key и Time to complete

Логические блоки 2 и 3 реализованы в соответствующих методах класса issue, логика которых была описана в предыдущем параграфе.

### 2.5.6. Написание velocity шаблона

Velocity шаблоны в плагинах для системы управления проектами «Atlassian Jira» используются для отображения результатов работы плагина. В Листинге 11 приведен пример кода из файла success.vm, который отображает 3 переменные из java класса реализатора на экран.

|  |
| --- |
| Колличество задач: $index <br>  Максимальный уровень вложенности подзадачи: $maxlevel <br>  Шкала: $scale <br> |

Листинг 11 success.vm, код отображения на странице переменных index, maxlevel и scale

Помимо простого отображения переменных, технология Velocity так же позволяет отображать элементы некоторого класса, а так же использовать различные условия, в Листинг 12 приведен пример отображения всех элементов класса задача, у которых parent\_key совпадает с полем key задачи родителя.

|  |
| --- |
| #foreach($issue3 in $issues)  #if ($issue3.getIssueParentkey() == $issue2.getIssueKey())  <li><h3><span>Подзадача $issue3.getIssueKey(), $issue3.getIssueName(), критерий $issue3.getIssueContribution()(+)</span></h3>  <ul>  <li>Описание: $issue3.getIssueDesc()</li>  <li>Приоритет: $issue3.getIssuePriority()</li>  <li>Сложность: $issue3.getIssueTime\_to\_complete()</li>  <li>Срочность: $issue3.getIssueUrgency()</li>  <li>Процент завершения: $issue3.getIssuePercantage\_of\_completion()%</li>  <li>Вклад: $issue3.getIssueWeight()</li>  <li>Значение: $issue3.getIssueValue()</li>  <li>Родитель: $issue3.getIssueParentkey()</li>  <li>Уровень: $issue3.getIssueLevel()</li>  <li>Осталось времени до завершения задачи(часы): $issue3.getIssueTime\_to\_end()h.</li>  </ul>  </li>  #end  #end |

Листинг 12 success.vm, код отображения всех подзадач задачи с IssueKey = $issue2.getIssueKey()

Отображение данных в velocity шаблоне плагина «Принятие решений» реализовано в следующем порядке: сначала выводится общая информация по проекту, это поля «количество задач», «максимальный уровень вложенности подзадачи» и «шкала», а затем выводятся в виде раскрывающегося списка все задачи 0 уровня, то есть те задачи, у которых нет задач родителей, для каждой задачи в начальном состоянии выводится 3 базовых поля, это ключ задачи, название задачи и критерий «Принятие решений», чем он выше, тем важнее заняться данной задачей именно сейчас. При нажатии на задачу 0 уровня, она раскрывается и на экране отображаются дополнительные поля задачи: описание, приоритет, сложность, срочность, процент завершения, вклад, значение, родитель, уровень, осталось времени до завершения задачи(часы), а так же поле подзадачи, при нажатии на которое отобразиться список всех подзадач данной задачи, их вывод отображается аналогично задачам 0 уровня. После вывода списка со всеми задачами идет блок графиков, но он уже реализован на основе технологии JavaScript, которая будет рассмотрена в следующем параграфе.

### 2.5.7. Использование JavaScript для вывода графиков

Для отображения графиков, демонстрирующих значение критерия «Принятие решений» по уровням вложенности задач используется скрипт google jsapi, в листинге 13 приведено описание такого графика для задач 0 уровня.

|  |
| --- |
| <script src="https://www.google.com/jsapi"></script>  <script>  google.load("visualization", "1", {packages:["corechart"]});  google.setOnLoadCallback(drawChart);  function drawChart() {  var data = google.visualization.arrayToDataTable([  ['Задача', 'Критерий']  #foreach($issue in $issues)  #if ($issue.getIssueParentkey() == "No parent")  ,['$issue.getIssueKey()', $issue.getIssueContribution()]  #end  #end  ]);  var options = {  title: '0 уровень',  hAxis: {title: 'Критерий'},  vAxis: {title: 'Задачи'}  };  var chart = new google.visualization.ColumnChart(document.getElementById('issue\_gist\_0'));  chart.draw(data, options);  }  </script>  <div id="issue\_gist\_0" style="width: 500px; height: 400px;"></div> |

Листинг 13 success.vm, код отображения графика распределения критерия «Принятие решений» по задачам 0 уровня

## 2.6. Результаты

В данном параграфе будут представлены результаты работа плагина «Принятие решений». На Рис. 28 изображено стартовое состояние плагина, который видит пользователь после перехода на страницу выполнения плагина, а конкретно это статистическая информация по проекту, которая включает в себя количество задач, максимальный уровень вложенности подзадач, а так же шкалу, а так же список всех задач 0 уровня и график, показывающий распределение критерия «Принятие решений» по всем задачам 0 уровня.

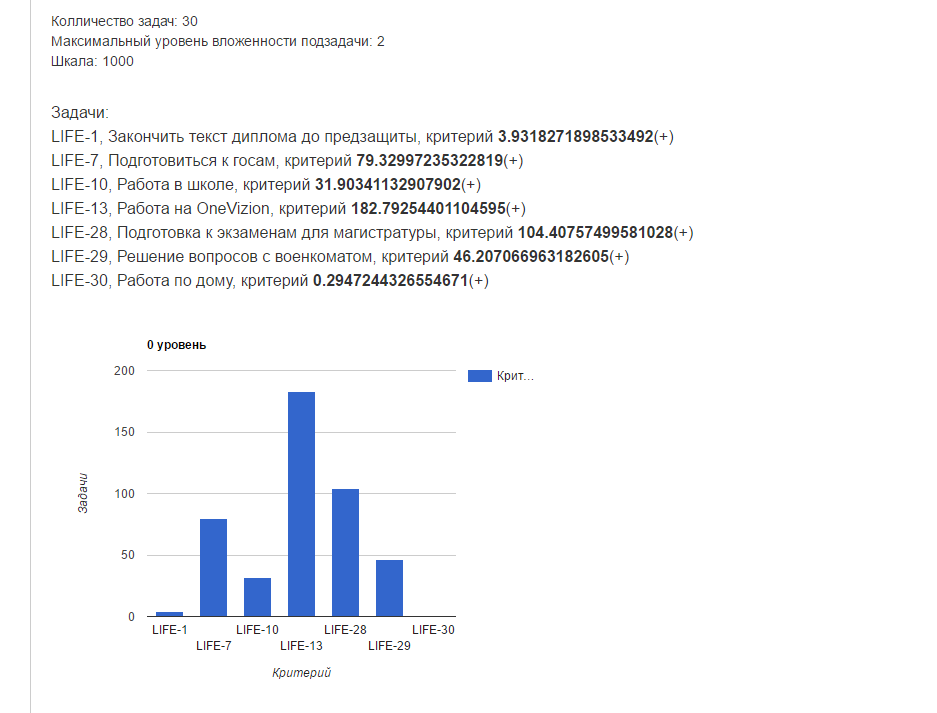


Рис. 28 Начальное состояние плагина «Принятие решений»

На Рис. 29 продемонстрирована подробная информация о 13 задаче проекта «LIFE», которая становится доступной при нажатии на данную задачу, она включает в себя поля: описание, процент завершения, уровень вложенности задачи, а так же список всех подзадач. Данный уровень может быть полезен для руководителей высокого уровня, которым важна ситуация в целом по проекту и не интересны подробности.

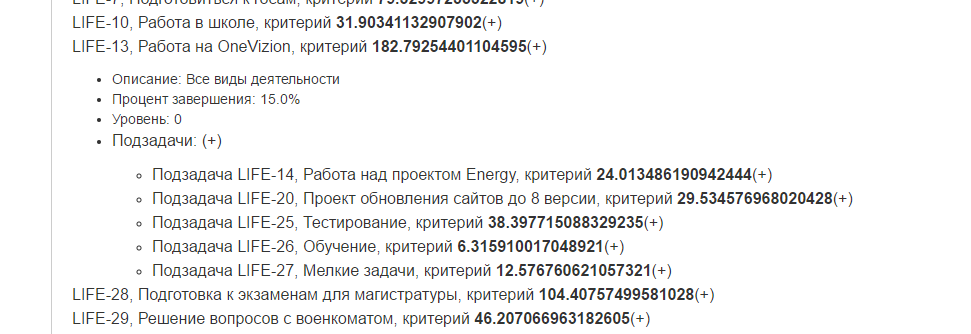


Рис. 29 Подробная информация о задаче LIFE-13

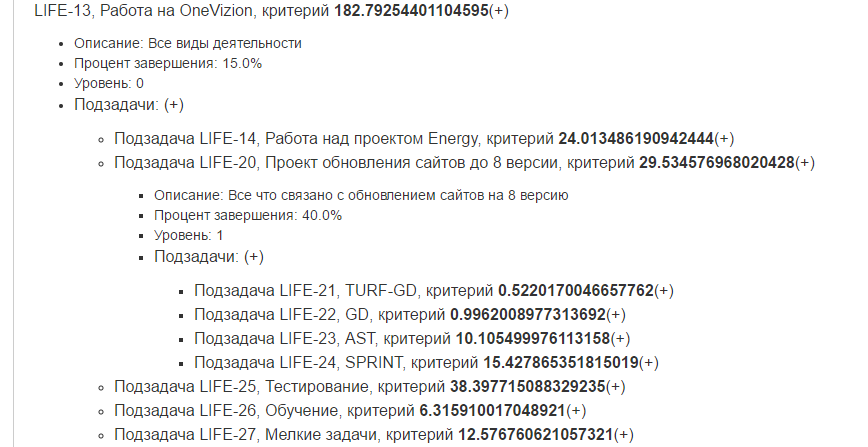
На Рис. 30 изображена подробная информация по подзадаче «LIFE-20» задачи «LIFE-13», которая включает в себя все те же поля, которые включает в себя задача «LIFE-13». Данный уровень может быть интересен руководителям более низкого уровня, которые управляют частями проекта.

Рис. 30 Подробная информация по подзадаче LIFE-20

На Рис. 31 изображена подробная информация по подзадаче «LIFE-20» задачи «LIFE-13», а так же подробная информация по подзадачи «LIFE-21» задачи «LIFE-20». Данный уровень может быть полезен непосредственно для исполнителей небольших задач.

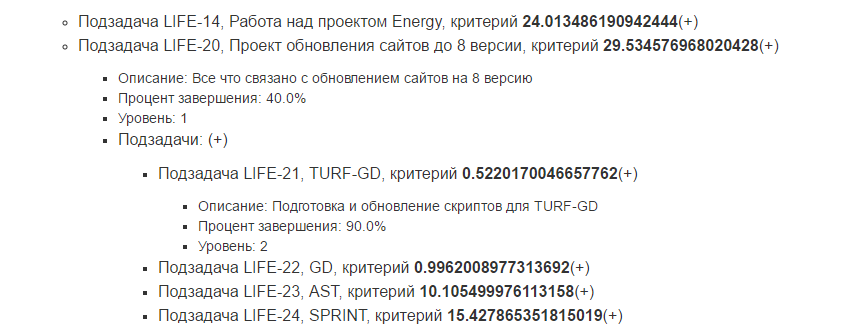


Рис. 31 Подробная информация по подзадачами LIFE-20 и LIFE-21

На Рис. 32 изображены графики распределения критерия «Принятие решений» по задачам 1 и 2 уровня вложенности задач.

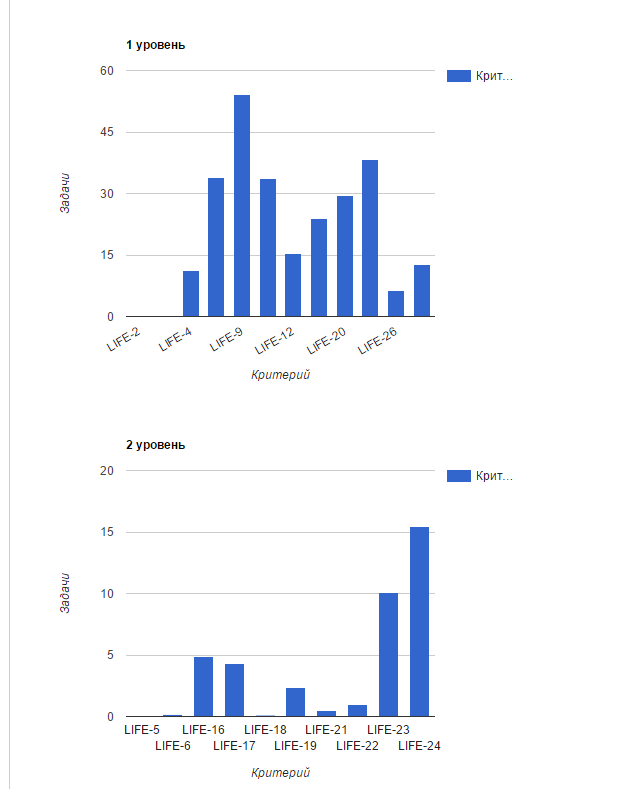


Рис. 32 Графики распределения критерия «Принятие решений» по задачам 1 и 2 уровня

# Заключение

В рамках выпускной квалификационной работы был выполнен литературный обзор информации и технологий необходимых для решения проблемы, а так же было разработано 2 различных критерия принятия решений. Кроме того в результате выполнения выпускной квалификационной работы был проведен анализ существующих средств решения проблемы менеджмента рабочего времени и способов графического отображения информации о проектах. В ходе данного анализа был выявлен ряд проблем общий для всех систем управления проектами, которые были в дальнейшем решены в данной работе, так же на основе анализа для реализации была выбрана система управления проектами «JIRA», так как она легко расширяема с помощью независимых программных модулей – плагинов, для графического отображения информации были выбраны деревья, так как они лучше всего подходят для отображения и анализа больших и сложно структурированных проектов. Кроме того была изучена технология плагинов для системы управления проектами «JIRA», в том числе с возможностью доступа к внешним источникам данных, для конечной версии плагина был использован способ получения данных с помощью стороннего плагина NFEED, так как он оказался наиболее удобным и эффективным для целей плагина. И в итоге на основе этих знаний был спроектирован и разработан плагин «Принятие решений» для программной системы управления проектами «JIRA». Данный плагин соответствует всем поставленным требованиям, преследуемая в работе цель достигнута.

Основные результаты данной работы была представлены и опубликованы на XII (XLIV) Международной научной конференции студентов и молодых ученых «Образование, наука, инновации: вклад молодых исследователей» в Кемеровском государственном университете, название публикации «Плагины в системе управления проектами «JIRA»».

# Литература

1. Официальный сайт проекта «Todoist» [Электронный ресурс]. – Режим доступа: https://ru.todoist.com/, свободный (дата обращения: 2.06.17).
2. Официальный сайт проекта «Any.do» [Электронный ресурс]. – Режим доступа: https://web.any.do/, свободный (дата обращения: 2.06.17).
3. Официальный сайт проекта «GanttPro» [Электронный ресурс]. – Режим доступа: https://ganttpro.com/ru/, свободный (дата обращения: 2.06.17).
4. Официальный сайт проекта «Asana» [Электронный ресурс]. – Режим доступа: https://asana.com/, свободный (дата обращения: 2.06.17).
5. Официальный сайт проекта «Easy Redmine» [Электронный ресурс]. – Режим доступа: <https://www.easyredmine.com/ru/>, свободный (дата обращения: 2.06.17).
6. Официальный сайт проекта «Freedcamp» [Электронный ресурс]. – Режим доступа: https://freedcamp.com/, свободный (дата обращения: 2.06.17).
7. Официальный сайт проекта «Bitrix24» [Электронный ресурс]. – Режим доступа: <https://www.bitrix24.ru/>, свободный (дата обращения: 2.06.17).
8. Официальный сайт проекта «Podio» [Электронный ресурс]. – Режим доступа: https://podio.com/, свободный (дата обращения: 2.06.17).
9. Официальный сайт проекта «Wrike» [Электронный ресурс]. – Режим доступа: https://www.wrike.com/ru/, свободный (дата обращения: 2.06.17).
10. Википедия, информация о диаграммах «Гантта» [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%93%D0%B0%D0%BD%D1%82%D1%82%D0%B0>, свободный (дата обращения: 2.06.17).
11. Википедия, информация о ментальных картах [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D1%81%D0%B2%D1%8F%D0%B7%D0%B5%D0%B9>, свободный (дата обращения: 2.06.17).
12. Википедия, информация о древовидных графах [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)>, свободный (дата обращения: 2.06.17).
13. Информация о проектах системы управления проектами «JIRA» [Электронный ресурс]. – Режим доступа: <https://confluence.atlassian.com/jira064/what-is-a-project-720416135.html>, свободный (дата обращения: 2.06.17).
14. Информация о задачах системы управления проектами «JIRA» [Электронный ресурс]. – Режим доступа: <https://confluence.atlassian.com/jira064/what-is-an-issue-720416138.html>, свободный (дата обращения: 2.06.17).
15. Википедия, информация о системе управления проектами «JIRA» [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Jira>, свободный (дата обращения: 2.06.17).
16. Русский официальный сайт проекта «Atlassian JIRA» [Электронный ресурс]. – Режим доступа: <https://ru.atlassian.com/software/jira>, свободный (дата обращения: 2.06.17).
17. Информация об архитектуре системы управления проектами «JIRA» [Электронный ресурс]. – Режим доступа: <https://developer.atlassian.com/jiradev/jira-platform/jira-architecture/jira-technical-overview>, свободный (дата обращения: 2.06.17).
18. Герберт Шилдт. Java 8. Полное руководство, 9-е издание = Java 8. The Complete Reference, 9th Edition. — М.: «Вильямс», 2015.
19. Matthew B. Doar. Practical Jira Plugins. — 2011.
20. Официальный сайт проекта «Velocity» [Электронный ресурс]. – Режим доступа: <http://velocity.apache.org/>, свободный (дата обращения: 2.06.17).
21. JavaScript: The Definitive Guide 1996 г. Автор: Дэвид Флэнаган.
22. Информация о построении графиков и диаграмм с помощью JavaScript [Электронный ресурс]. – Режим доступа: <http://htmlbook.ru/blog/grafiki-i-diagrammy>, свободный (дата обращения: 2.06.17).
23. Документация по JIRA Java API [Электронный ресурс]. – Режим доступа: <https://developer.atlassian.com/docs/>, свободный (дата обращения: 2.06.17).
24. Хабрахабр, пример разработки плагина для системы управления проектами «JIRA» [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/company/dataart/blog/273429/>, свободный (дата обращения: 2.06.17).
25. Официальный сайт проекта «Tomcat» [Электронный ресурс]. – Режим доступа: <https://tomcat.apache.org>, свободный (дата обращения: 2.06.17).
26. Информация об установке «Java» [Электронный ресурс]. – Режим доступа: <https://www.java.com/ru/download/help/download_options.xml>, свободный (дата обращения: 2.06.17).
27. Информация об установке «Atlassian SDK» [Электронный ресурс]. – Режим доступа: <https://developer.atlassian.com/docs/getting-started/set-up-the-atlassian-plugin-sdk-and-build-a-project/install-the-atlassian-sdk-on-a-windows-system>, свободный (дата обращения: 2.06.17).
28. Информация о модулях плагина системы управления проектами «JIRA» [Электронный ресурс]. – Режим доступа: <https://developer.atlassian.com/docs/getting-started/plugin-modules>, свободный (дата обращения: 2.06.17).
29. Официальный сайт проекта «NetBeans» [Электронный ресурс]. – Режим доступа: <https://netbeans.org/>, свободный (дата обращения: 2.06.17).
30. Официальный сайт проекта «NotePad++» [Электронный ресурс]. – Режим доступа: <https://notepad-plus-plus.org/>, свободный (дата обращения: 2.06.17).
31. Официальный сайт проекта «MySQL» [Электронный ресурс]. – Режим доступа: <https://www.mysql.com/downloads/>, свободный (дата обращения: 2.06.17).
32. Официальный сайт проекта «MySQL workbench» [Электронный ресурс]. – Режим доступа: <https://www.mysql.com/products/workbench/>, свободный (дата обращения: 2.06.17).
33. Плагин «NFEED» в официальном магазине «Atlassian» [Электронный ресурс]. – Режим доступа: <https://marketplace.atlassian.com/plugins/com.valiantys.jira.plugins.SQLFeed/server/overview>, свободный (дата обращения: 2.06.17).
34. Официальный сайт проекта «NFEED», помощь и поддержка [Электронный ресурс]. – Режим доступа: <https://valiantys.atlassian.net/wiki/display/NFEED/nFeed+Getting+Help+and+Support>, свободный (дата обращения: 2.06.17).
35. Википедия, информация о плагинах [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD>, свободный (дата обращения: 2.06.17).
36. Информация о подключении системы управления проектами «JIRA» к базе данных MySQL [Электронный ресурс]. – Режим доступа: <https://confluence.atlassian.com/adminjiraserver073/connecting-jira-applications-to-mysql-861253043.html>, свободный (дата обращения: 2.06.17).
37. Информация о подключении системы управления проектами «JIRA» к внешними источнику данных с помощью «JNDI» соединения [Электронный ресурс]. – Режим доступа: <https://confluence.atlassian.com/jirakb/how-to-connect-to-postgresql-using-jndi-720418521.html>, свободный (дата обращения: 2.06.17).

# Приложение, содержание прилагаемого CD-диска

1. jira\_plugin\_didenko.docx – текст выпускной квалификационной работы.
2. jira\_plugin\_didenko.doc – текст выпускной квалификационной работы.
3. jira\_plugin\_didenko.pdf – текст выпускной квалификационной работы.
4. jira\_plugin\_didenko.pptx – презентация выпускной квалификационной работы.
5. jira\_plugin\_didenko.rar – исходный код плагина «Принятие решений».
6. jira\_plugin\_didenko\_instruction.doc – инструкции по работе с плагином «Принятие решений».