**Курсовая работа:**

1. Обоснование дипломного проекта – первая часть пояснительной записки (начать к концу сентября):

* Описание темы (что под ней подразумевается) – постановка задачи;
* Исследование предметной области (анализ проблематики; задача, которую мы решаем нашим дипломом) – определение методов, уточнение требований к программному модулю – зачем, почему, для кого, предпосылки;
* Обзор аналогов и выводы из обзора (их описание, обоснование и сравнение);
* Формирование функциональных требований к дипломному проекту.

1. Функциональные требования (выбор технических средств разработки).
2. Модель будущего дипломного проекта (диаграммы, алгоритмы).
3. Техническое задание.
4. Разработка дипломного проекта (март).
5. Техническое описание.
6. Экономическая оценка дипломного проекта – экономика отрасли.
7. Пояснительная записка (2-3 приложения: Модель, ТЗ, Тех. Описание) – около 70 страниц.

Затем получаем отзыв от заказчика (лицензия) и руководителя.

**Защита**: документы + презентация.

**Мастер-классы** от экспертов и внутренних лиц по презентации своего диплома.  
Расписать к 19 сентября:

Описание бизнес процесса, выделить предпосылки, слабые стороны, и чем именно моя работа будет их компенсировать. Функционал – начальные требования.

Формулировка поставленной задачи

Задачей является функциональное моделирование приложения для управления (развернуть) проектами по разработке ПО (далее - система), которое будет обеспечивать пользователей возможностью распределения задач в рамках проекта, управления сроками выполнения задач, прикреплением файлов с данными, относящимися к проекту, а также визуализации (исходных требований к проекту) задач посредствам ассоциативных карт.

Назначение разработки – обеспечение пользователей системой для распределения задач, а также контроля за выполнением задач при реализации проекта.

Ресурс нацелен, в первую очередь, на команды, выполняющих разработку программного обеспечения.

Процессы разработки программного обеспечения и администрирования проекта сопряжены с рядом проблем (особенностей, трудностей ):

* Дополнительная нагрузка на руководителя команды в связи с уточнением задачи для разработчиков;
* сложность с распределением задач по разработчикам;
* контроль за соблюдением разработчиками сроков выполнения поставленных задач;
* сложность полной постановки задачи без использования дополнительных материалов;
* проблемы с поиском корректной дополнительной информации для разработчиков;
* малый опыт разработки программистами в команде;
* проблемы с передачей информации о проекте удаленным разработчикам;
* сложность отображения сроков выполнения задач на календарь в связи с нерациональной тратой времени;

Разрабатываемый ресурс нацелен на решение этих проблем.

обзор предметной области

(Сценарий выполнения проекта, есть руководитель, исполнители)

(Расписал сегодня, так как сказали переделать)

Процессы разработки программного обеспечения, администрирования проекта сопряжены с определенными особенностями.

(Пример) При разработке программного обеспечения выделяют команду разработчиков и руководителя проекта.

При планировании стадий реализации проекта руководитель должен: во-первых, создать задачи для команды разработчиков, во-вторых, распределить созданные задачи по разработчикам. Также, необходимо качественно проработать сроки выполнения поставленных задач, обеспечить разработчиков четко поставленной задачей и необходимыми ресурсами, которые могут обеспечить достоверной и полной информацией по конкретной теме.

В ходе выполнения задач может потребоваться связь между руководителем проекта (менеджером) и разработчиком (дизайнером?) для разрешения возникающих вопросов, технических или организационных моментов.

(Старая часть, которую не менял)

При планировании проекта менеджер должен обеспечить распределение стадий разработки по участникам команды таким образом, чтобы осуществлялся намеченный жизненный цикл проекта. Также, необходимо качественно проработать сроки выполнения поставленных задач и обеспечить разработчиков четко поставленной задачей, а также необходимыми ресурсами, которые могут обеспечить достоверной и полной информацией по конкретной теме.

При процессе распределения задач по разработчикам перед руководителем команды возникают следующие сложности: необходимо подробно объяснить каждому программисту поставленную задачу, а также отвечать на вопросы, возникающие в процессе написания кода программы.

При участии в процессе разработки у начинающих разработчиков возникают проблемы с отсутствием опыта командной разработки, в том числе при поиске дополнительной информации в интернете возникает проблема ложности информации из-за большого количества людей, не достаточно компетентных в данной области, в связи с чем снижается скорость разработки ПО.

В случае участия в проекте удаленных разработчиков возникает проблема взаимодействия между участниками команды, а также обеспечение разработчика необходимой информацией.

Проблемы, описанные выше могут быть решены с помощью разрабатываемой системы:

При создании изначальной структуры списка выполняемых задач присутствует возможность составления ассоциативной карты, которая может быть преобразована в доску.

При создании доски (проекта) есть возможность создать колонки (группы задач), а в колонках – создать карточки, в которых описывается конкретная задача.

Создание колонок и карточек решает проблемы распределения задач, а также сложности передачи информации о проекте удаленным разработчикам.

Описание карточки содержит следующие части:

* Блок с описанием поставленных задач (решает проблемы, связанные с дополнительной информацией для разработчиков);
* Чек-лист для задач;
* Срок выполнения данной карточки (решает сложности, связанные с контролем сроков выполнения задач);
* Люди, прикрепленные к карточке (решает проблемы с командной разработкой);
* Блок с вложениями (решает проблему поиска корректной информации);
* Блок с комментариями (решает проблему уточнения задач руководителем команды);

Также, доступно добавление к карточкам меток, дифференцирующих карточки по типу задачи.

выбор модели жизненного цикла

При выборе модели жизненного цикла программного обеспечения, подразумевается, что при разработке проекта есть команда, состоящая из следующих лиц:

1. Менеджер проекта.
2. Проектировщик подсистем.
3. Разработчики.
4. Руководитель команды.
5. Тестировщик.

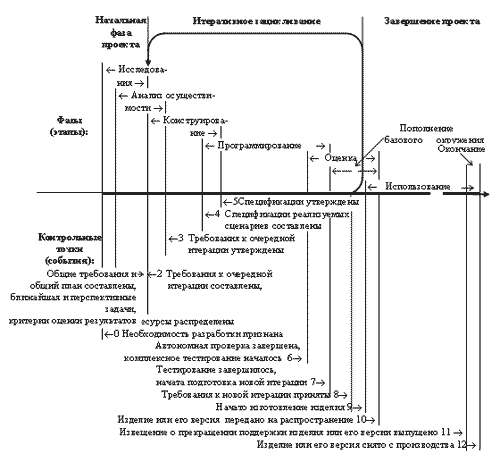
Данная модель жизненного цикла имеет ряд преимуществ:

* При разработке ПО фазы проекта могут выполняться параллельно, что уменьшает время реализации;
* Продукт может реализоваться в несколько этапов: сначала основной функционал программы, затем наращивание дополнительного;
* Существуют контрольные точки каждого из фаз разработки проекта, что уменьшает возможные просчеты в момент перехода на следующие фазы разработки;

Разработка системы планируется в несколько этапов: сначало внедрение базового функционала, затем постепенное наращивание дополнительного. Без базового функционала отсутствует возможность разрабатывать дополнительный, так как при просчетах в базовом функционале системы необходимо перестраивать всю систему.

Данная модель жизненного цикла подходит для реализации системы для управления проектами, так как разработка функционала программы по этапам наиболее оптимальна.

Исходя из того, что есть команда разработчиков, а также, что при разработке будет использоваться объектно-ориентированный подход, выберем объектно-ориентированную модель жизненного цикла программного обеспечения (Рисунок 1).



*Рисунок 1 Модель жизненного цикла ПО*

Сравнение аналогов

В ходе обзора аналогов были выбраны следующие решения:

1. «Trello».
2. «Asana».
3. «Jira».
4. «Mindmeister».
5. «Mindmup».

«Trello» -- прямой аналог. Это разработка компании «Fog Creek Software», предназначенная для управления проектами небольших групп. Проекты представлены в виде «досок», в которых есть разбиение на столбцы, в которых содержатся карточки с задачами.

Отличительной особенностью «Trello» является возможность расширения функционала путем интегрирования сторонних приложений в разрабатываемый проект.

Этот аналог не подходит для решения задачи, поставленной перед проектируемым ПО в силу того, что отсутствует возможность представления данных в виде ассоциативной карты. Вследствие чего, отсутствует возможность отображения задач при «мозговом штурме» без использования сторонних приложений.

«Asana» -- прямой аналог. Это разработка компании «Рога и копыта», мобильное и веб-приложение для управления проектами в небольших командах. Приложение представлено в виде списка задач, которые прикреплены к проекту.

Отличительной особенностью «Asana» является возможность взаимодействия с календарем посредством отображения на календарь сроков выполнения задач, а также их создания.

Этот аналог не подходит для решения задачи, поставленной перед проектируемым ПО в силу того, что отсутствует возможность представления данных в виде ассоциативной карты. Вследствие чего, отсутствует возможность отображения задач при «мозговом штурме» без использования сторонних приложений.

«Jira» -- прямой аналог. Это разработка компании «Atlassian», коммерческая система отслеживания ошибок, предназначена для организации взаимодействия с пользователями, хотя в некоторых случаях используется и для управления проектами.

Отличительной особенностью «Jira» является возможность создания различных отчетов, отображающих состояние выполнения проекта.

Этот аналог не подходит для решения задачи, поставленной перед проектируемым ПО в силу того, что отсутствует возможность представления данных в виде ассоциативной карты. Вследствие чего отсутствует возможность отображения задач при «мозговом штурме» без использования сторонних приложений.

В силу того, что рассматриваемые прямые аналоги схожи по функционалу, выделим основные функции представленных систем для управления проектами.

Перечень общих функций рассмотренных продуктов:

1. Создание карточек, а также возможность их детального описания: добавление описания карточки, а также добавления списков задач.
2. Прикрепление файлов и ссылок к карточкам. К каждой карточке можно прикрепить файлы в различных форматах или ссылки на сторонние ресурсы.
3. Добавление комментариев к карточкам в текстовом формате, а также прикрепить ссылку на сторонний ресурс.
4. Установка и контроль за сроками выполнения задач. Для каждой карточки можно установить срок выполнения, а также проконтролировать его соблюдение.
5. Добавление участников к разработке проекта. К карточкам можно прикреплять одного или нескольких участников, которые будут отображаться в проекте.

«Mindmeister» -- косвенный аналог. Это разработка компании «MeisterLabs GmbH», предназначенная для майндмэппинга.

Этот аналог предназначен для решения следующей задачи: представления данных в виде ассоциативной карты.

«Mindmup» -- косвенный аналог. Это разработка компании «Sauf Pompiers Ltd», предназначенная для создания майнд карт.

Этот аналог предназначен для решения следующей задачи: представления данных в виде ассоциативной карты.

В силу того, что рассматриваемые косвенные аналоги схожи по функционалу, выделим основные функции представленных аналогов.

Перечень общих функций рассмотренных продуктов:

1. Создание майнд-карт: отображение данных в виде деревьев для более эффективного восприятия информации.
2. Поиск по веткам: среди всех веток можно найти концевой узел или подветку по заданному значению.
3. Добавление участников к построению карты. К картам можно прикреплять одного или нескольких участников, которые будут отображаться в проекте.
4. Добавление текстовых заметок к рабочей области. К рабочей области можно прикрепить заметки с текстом, которые объясняют или поясняют информацию на карте.
5. Прикрепление файлов к карте. К карте можно прикрепить файлы в различных форматах.

Сравнение функций представленных аналогов приведено в таблице. При оценке используются следующие критерии:

* Командная разработка – возможность участия в проекте нескольких людей одновременно;
* Прикрепление файлов – возможность прикрепления дополнительных материалов (файлы, ссылки) к задачам;
* Платформы – какие платформы (операционные системы) поддерживает приложение;
* Стоимость – размер платы (в рублях) за использование продукта в месяц;
* Поиск по данным – возможность фильтрации информации по средствам поиска по имеющимся данным;
* Установка сроков выполнения – возможность установки сроков выполнения задач (дата и время, которые обозначают крайний срок выполнения);
* Представление данных в виде майнд-карт – возможность представления дынных в виде майнд-карт;
* Создание команд – возможность создания команд, объединяющих пользователей для совместной работы(дополнить);
* Отображение сроков на календарь – возможность отображения сроков выполнения в виде календаря с метками задач;
* Создание диаграмм – возможность создания графического представления данных в виде диаграмм;
* Авторизация – необходимость предварительной регистрации в приложении для его использования;

Таблица 1 – Сравнение функционала аналогов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Прямые аналоги | | | Косвенные аналоги | |
| Trello | Asana | Jira | Mindmeister | Mindmup |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Командная разработка | + | + | + | + | + |
| Прикрепление файлов | + | + | + | + | + |
| Платформы | Windows, MacOS, iOS, Android, Web | iOS, Android, Web | Windows, Linux, Web | iOS, Android, Web | Web |
| Стоимость | 700 р/месяц | 700 р/месяц | 700 р/месяц | 550 р/месяц | 200 р/месяц |
| Поиск по данным | + | + | + | + | + |
| Установка сроков выполнения | + | + | + | - | - |
| Представление данных в виде майнд карт | - | - | - | + | + |
| Создание команд | + | + | + | - | - |
| Отображение сроков на календарь | - | + | + | - | - |
| Создание диаграмм | - | - | + | + | + |
| Авторизация | + | + | + | + | + |

В ходе обзора аналогов был сделан вывод, что в качестве образца для проектируемого ПО можно выделить «Trello», потому что в нем данные представляются наиболее удобно, каждый проект разделен на направления(столбцы), а сами направления разделены на задачи(карточки), в которых присутствует подробное описание задачи и сроков ее выполнения. Также, в данном аналоге присутствует возможность расширения функционала, что способствует более детальному описанию проекта.

функциональные требования

В ходе обзора аналогов были выявлены следующие функции, которые необходимо в проектируемом ПО (поменять порядок, выделить более глобальные функции):

1. Авторизация.
2. Создание карточек, а также возможность их детального описания: добавление описания карточки, а также добавления списков задач.
3. Установка и контроль за сроками выполнения задач. Для каждой карточки можно установить срок выполнения, а также проконтролировать его соблюдение.
4. Создание команд, а также командных досок.
5. Добавление участников к командным доскам.
6. Создание майнд-карт: отображение данных в виде деревьев для более эффективного восприятия информации.

Помимо функций, описанных выше, необходимо реализовать так же следующие функции:

1. Сортировка по типу карточек.
2. Преобразование майн карт в список задач. Созданная майнд-карта автоматически конвертируется в список задач.
3. Редактирование майнд-карты.
4. Добавление комментариев к карточкам.
5. Добавление чек-листов к карточкам.
6. Добавление к карточкам описания в виде текстового поля.
7. Отображение всех карточек конкретного участника.
8. Добавление тэгов к карточке.

Анализ рисков

В ходе разработки могут возникнуть следующие риски:

1. Увеличение требований к продукту со стороны заказчика. Добавление функциональных или внедрение правок в существующий проект.

Возникновение этой ситуации может привести к следующим последствиям: увеличение сроков, необходимых для реализации поставленных задач, вследствие чего увеличивается стоимость выполнения, также внесение дополнительных функциональных требований может потребовать полного перепроектирования продукта.

Чтобы избежать возникновения этой ситуации, необходимо заключить договор с заказчиком, в котором фиксируется ТЗ.

В случае возникновения этой ситуации, необходимо уведомить заказчика о том, что вся ответственность за срыв сроков и увеличение стоимости проекта лежит на нем.

1. Увольнение опытного сотрудника, который участвовал в разработке проекте.

Возникновение этой ситуации может привести к следующим последствиям: невозможность разработки проекта, срыв сроков выполнения задач, вследствие чего финансовые убытки.

Чтобы избежать возникновения этой ситуации, необходимо подписать контракт с сотрудниками о проектной работе.

В случае возникновения этой ситуации, необходимо перераспределить обязанности между оставшимися сотрудниками, параллельно искать сотрудника на данную должность, за счет чего уменьшится влияние на сроки и на стоимость проекта.

1. Несоблюдение сроков реализации проекта .

Возникновение этой ситуации может привести к следующим последствиям: потеря финансов из-за санкций ввиду срывов сроков реализации, потеря репутации, увеличение нагрузки на команду.

Чтобы избежать возникновения этой ситуации, необходимо обсудить техническое задание с командой, составить план разработки и тестирования, определить критерии сдачи - приемки проекта, а также заложить планируемые выходные и праздничные дни, отпуска сотрудников и получить предполагаемую дату сдачи проекта.

В случае возникновения этой ситуации, необходимо увеличить темпы разработки продукта, перераспределить нагрузку в случае наличия менее нагруженных сотрудников.

1. Низкая производительность команды в следствие каких-либо внешних факторов

Возникновение этой ситуации может привести к следующим последствиям: нарушение сроков выполнения задач, а также к разладу команды, что ведет к экономическим убыткам.

Чтобы избежать возникновения этой ситуации, необходимо ввести контрольные точки выполнения проекта с точными сроками выполнения работ.

В случае возникновения этой ситуации, необходимо реализовать основной функционал проекта, а после - постепенная доработка и увеличение функционала. Также проведение мероприятий по повышению продуктивности разработчиков.

1. Возникновение сложности при реализации проекта ( невозможность реализации функции или части проекта ввиду сложности задачи ).

Возникновение этой ситуации может привести к следующим последствиям: нарушение сроков выполнения задач, ухудшение отношений внутри коллектива, что ведет к снижению производительности, остановка реализации проекта, как следствие - финансовые убытки.

Чтобы избежать возникновения этой ситуации, необходимо детально продумать план выполнения задач, а также сделать детальный анализ ТЗ.

В случае возникновения этой ситуации, необходимо нанять разработчиков, которые помогут решить сложности, возникшие в процессе разработки.

1. Выход из строя технического оборудования

Возникновение этой ситуации может привести к следующим последствиям: остановка проекта, нарушение сроков выполнения задач.

Избежать возникновения данной ситуации невозможно, можно лишь уменьшить возможные потери путем контроля за техническим оборудованием, а также использовать резервные копии.

В случае возникновения этой ситуации, необходимо заменить вышедшую из строя технику.

1. Нарушение авторских прав. Использование разработчиком без ведома проектного менеджера чужого исходного кода или алгоритма.

Возникновение этой ситуации может привести к следующим последствиям: иски в суд, потеря времени на урегулирование конфликта, появление уязвимостей в разрабатываемой системе.

Чтобы избежать возникновения этой ситуации, необходимо обязать разработчиков обсуждать с менеджером проекта возможность легального использования чужого кода или алгоритма.

В случае возникновения этой ситуации необходимо минимизировать последствия путем договора с автором исходного кода или алгоритма.

Информация о приоритетности рисков приведена в таблице 2.

Таблица 2 – Приоритетность рисков

|  |  |  |  |
| --- | --- | --- | --- |
| Риски | Вероятность наступления риска (от 1 до 10) | Степень влияния риска на проект (от 1 до 10) | Приоритет риска (от 1 до 100) |
| Увеличение требований к продукту со стороны заказчика. | 5,3 | 9 | 47,7 |
| Увольнение опытного сотрудника | 2,9 | 9,4 | 89 |
| Несоблюдение сроков реализации проекта | 6,1 | 6,7 | 70 |
| Низкая производительность команды | 6,7 | 7,8 | 52,2 |
| Возникновение сложностей при реализации проекта | 7,1 | 5,2 | 46,8 |
| Выход из строя технического оборудования | 3,9 | 6,9 | 27,8 |
| Нарушение авторских прав | 2,9 | 5,8 | 31,2 |