МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ОТЧЕТ

о прохождении производственной практики

на ИООО «ЭПАМ СИСТЕМЗ»

Студент гр. АСОИ-181 Остапенко А. К.

(подпись) (дата)

Руководитель от университета Широченко В. А.

(подпись) (дата)

Руководитель от предприятия Широченко В. А.

(подпись) (дата)

Могилев, 2022 г.

Содержание

[Введение 3](#_Toc101294229)

[Индивидуальное задание 4](#_Toc101294230)

[Заключение 8](#_Toc101294231)

[Список использованной литературы 9](#_Toc101294232)

# Введение

Преддипломная практика была пройдена на предприятии ИООО «ЭПАМ Системз».

Индивидуальное задание на практику было сформулировано следующим образом: изучить объект автоматизации, назначение, структуру и особенности функционирования; выявить недостатки и возможности повышения эффективности; осуществить постановку задачи дипломного проектирования; автоматизировать процесс взаимодействия команды разработчиков.

Во время прохождения технологической практики на предприятии, в мои должностные обязанности входило следующее:

* Соблюдение техники безопасности на всей территории офиса;
* Разработка программного модуля и самого приложения;
* Оформление необходимой технической документации для приложения;
* Освоение и применение в работе новых технологий;
* Проводить отладку разработанного приложения.

Для выполнения данного задания потребовалась следующая информация:

* Какие СУБД используются на предприятии;
* Какие поля необходимы для БД и корректного использования на предприятии в работе;
* Какие языки программирования предпочтительнее использовать в ходе работы.

Отчёт по преддипломной практике состоит из основных разделов: содержание, введение, индивидуальное задание, заключение и список использованной литературы.

# Индивидуальное задание

Тема индивидуального задания: «Автоматизация процесса взаимодействия команды разработчиков программного обеспечения».

Объектом автоматизации является процесс взаимодействия членов команды разработчиков.

Зачастую при разработке программного обеспечения члены команды сталкиваются со многими проблемами, которые необходимо решать для развития проекта. Например, во время создания очередной задачи нужно продумать: какие данные вносить для достаточно подробного описания задачи, в каком формате и по какому шаблону это делать. Ведь очень важно придерживаться некоторого общего шаблона, чтобы после создания задачи одним разработчиком, второму не пришлось собирать воедино всё, что сделал первый. Так же членам команды нужно придумать, как делиться новыми задачами, как обсуждать эти задачи, чтобы о результатах обсуждения могли узнать остальные разработчики. Само собой разумеется, что всем членам команды необходимо иметь возможность контролировать состояние каждой задачи в каждый момент времени, чтобы не начать работать над одной и той же задачей одновременно. При каком бы то ни было взаимодействии, важно понимать, какая роль у вашего собеседника. Часто приходится тратить время на то, чтобы найти человека с конкретной ролью на проекте: тестировщик, разработчик, менеджер и т. д. Очевидно, что всем участникам проекта необходимо обезопасить свою среду взаимодействия и результаты работы от внешних угроз.

При одновременной работе нескольких программистов задачи могут выполняться параллельно. В таком случае разные разработчики выполняют разные задачи. Однако они в силу различных причин тратят на выполнение задач неодинаковое количество времени.

Например, рассмотрим следующий граф (рисунок 1). Состояния 1 и 2 являются соответственно начальным и конечным состояниями некоторого разрабатываемого приложения в некотором отрезке времени. Для перехода из состояния 1 в состояние 2 необходимо выполнить все задачи, обозначенные буквами. В процессе параллельной разработки задачи A1, B1-2, C1-2 могут быть выполнены существенно быстрее задач A2, C3. В таком случае разработчики, закончившие раньше, начинают простаивать, что является неэффективным использованием времени.

Наше приложение устраняет причины лишних затрат времени.

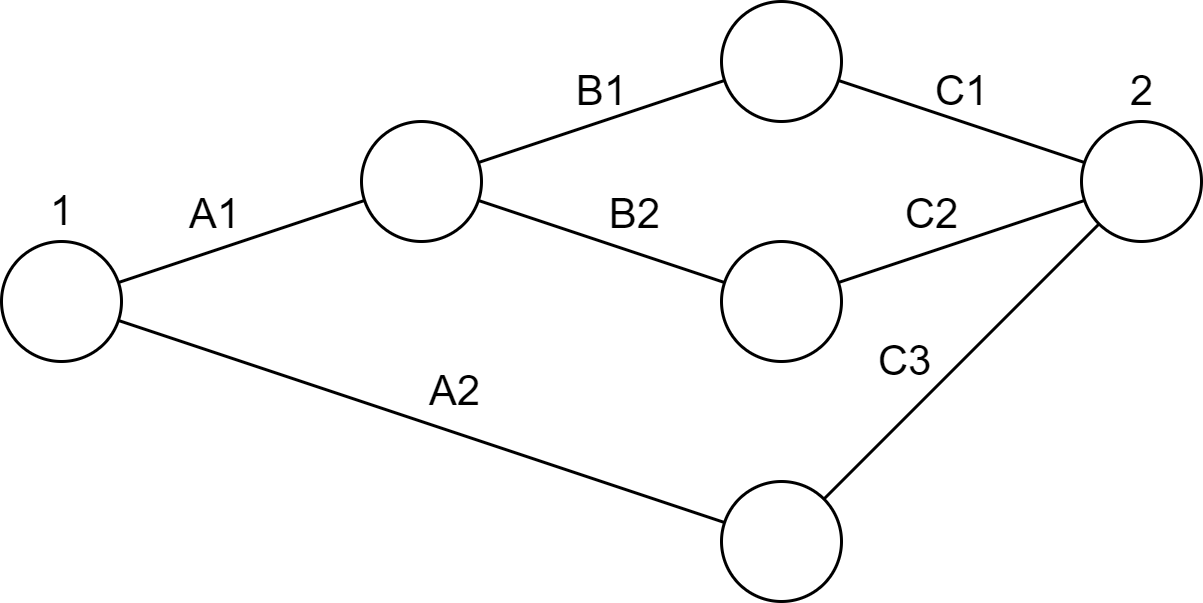


Рисунок 1 — Граф состояний проекта

Для начала введём понятие критического пути. Критический путь состоит из самой длительной последовательности задач от начала проекта до его завершения. Тогда время, которое необходимо затратить на разработку всего проекта (*T*), равно времени, затрачиваемом на каждую задачу, лежащую на критическом пути. Тогда:

где *t* — время, которое необходимо затратить на одну задачу,

*i* — номер задачи на критическом пути.

Каждая задача состоит из одинаковых этапов. Тогда *t* можно представить как:

где *tdes* — время на описание задачи,

*tdev* — время на непосредственно написание кода,

*tcoop* — время, затрачиваемое на взаимодействие членов команды.

Тогда *T* можно представить так:

Из формулы видно, что, уменьшая время на выполнение задач, лежащих на критическом пути, можно достичь сокращения времени на выполнение всего проекта.

Наше приложение сокращает переменную *tdes*, предоставляя шаблон заполнения данных о задаче со значениями по умолчанию.

Так же, например, наше приложение сокращает время на взаимодействие (*tcoop*), сразу отображая задачу всем членам команды при её создании. Таким образом пользователю не нужно затрачивать время на то, чтобы уведомить остальных членов команды о создании новой задачи и объяснить её суть.

При разработке программного обеспечения члены команды сталкиваются со многими проблемами, которые необходимо решать для развития проекта. Например, во время создания очередной задачи нужно продумать: какие данные вносить для достаточно подробного описания задачи, в каком формате и по какому шаблону это делать. Ведь очень важно придерживаться некоторого общего шаблона, чтобы после создания задачи одним разработчиком, второму не пришлось собирать воедино всё, что сделал первый. Руководителю команды разработчиков необходимо эффективно распределять ресурсы проекта. Для этого ему требуется иметь агрегированное визуальное представление всех задач с информацией о том, какую сложность имеет та или иная задача, как они взаимосвязаны между собой и как распределены члены команды между задачами. Так же членам команды нужно придумать, как делиться новыми задачами, как обсуждать эти задачи, чтобы о результатах обсуждения могли узнать остальные разработчики. Само собой разумеется, что всем членам команды необходимо иметь возможность контролировать состояние каждой задачи в каждый момент времени, чтобы не начать работать над одной и той же задачей одновременно. При каком бы то ни было взаимодействии, важно понимать, какая роль у вашего собеседника. Часто приходится тратить время на то, чтобы найти человека с конкретной ролью на проекте: тестировщик, разработчик, менеджер и т. д. Очевидно, что всем участникам проекта необходимо обезопасить свою среду взаимодействия и результаты работы от внешних угроз.

Данные проблемы можно решить следующим образом:

* Необходим механизм, позволяющий общаться между собой в реальном времени. Результаты общения должны быть видны всем членам команды.
* Необходим механизм оценки трудоёмкости задач.
* Необходим механизм визуализации связей между задачами.
* Необходим механизм полного контроля состояния задач.
* Необходим механизм, для стандартизации всех входных данных.
* Необходим механизм, позволяющий ознакомится с каждым членом команды. Так же должна быть возможность для связи со всеми участниками проекта.
* Необходим механизм авторизации и аутентификации.

Для решение всех вышеперечисленных проблем при разработке клиентской части необходимо будет создать следующие инструменты. Все члены команды смогут видеть задачи текущего проекта на специальной доске, где можно будет управлять состоянием каждой задачи. Так же, к каждой задаче можно будет оставлять комментарий, и любой разработчик сможет увидеть обсуждение. Будут разработаны специальные формы для того, чтобы ввод данных для всех участников имел общий вид. Никаким другим способом задачи создать будет нельзя. Так же многие формы создания объекта будут служить и для его изменения. Будет внедрён механизм кэширования часто используемых данных. Будет разработан механизм, позволяющий присвоить каждой задаче приблизительную оценку трудоёмкости. Будет внедрён механизм добавления связей между задачами, которые нужно выполнять последовательно друг за другом. Будет разработан механизм визуализации связей между задачами с отображением оценок трудоёмкости в виде графа. На графе будет выделяться цветом критический путь. Будет разработана специальная страница, где будет находиться таблица со всеми участниками проекта. На ней можно будет ознакомиться со списком всех разработчиков и их данными. На этой же странице будет предусмотрен механизм взаимодействия посредством электронной почты.

# Заключение

В процессе разработки ПО командами разработчиков может возникать множество проблем. Одна из главных проблем — ограниченное время разработки. Поэтому важно, чтобы время не было потрачено впустую. Для решения этой проблемы нами была составлена модель, описывающая процесс разработки ПО. Из неё видно, что благодаря правильной и удобной координации действий между членами команды разработчиков можно достигнуть более оптимального взаимодействия между ними. В частности, это касается распределения задач между членами команды. Наше приложение позволяет делать это эффективно и наглядно, что сокращает время, затрачиваемое на координацию членов команды.

# Список использованной литературы

1. Metanit Entity Framework Core [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/entityframeworkcore/1.1.php
2. Microsoft Docs Entity Framework Core [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/en-us/ef/core/
3. Грабер, Мартин SQL для простых смертных / Мартин Грабер. - М.: ЛОРИ, 2014. - 378 c.
4. Подбельский, В. В. Язык С#. Решение задач / В. В. Подбельский. – М.: Инфра-М, 2014. – 296 с.
5. Троелсен, Э. Язык программирования С# 4.0 и платформа .NET 4.0 : пер. с англ. / Э. Троелсен. – М. : Вильямс, 2011. – 1392 с.
6. Маркин, А. В. Построение запросов и программирование на SQL. Учебное пособие / А.В. Маркин. - М.: Диалог-Мифи, 2014. - 384 c.