TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа информационных технологий и робототехники (ИШИТР)
Направление подготовки 09.03.02 Информационные системы и технологии
Отделение школы (НОЦ) информационных технологий

# МАГИСТЕРСКАЯ РАБОТА

| Тема работы |
|---|
| **Разработка сервиса для проверки зависимостей сборок проектов с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps** |

УДК 004.744:004.415.2

Студент

| Группа | ФИО | Подпись | Дата |
|---|---|---|---|
| 8ИМ92Б | Смирнов Павел Олегович | | |

Руководитель ВКР

| Должность | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент отделения ИТ | Мирошниченко Евгений Александрович | К.т.н. | | |

## КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

| Должность | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ОСГН ШБИП | Верховская Марина Витальевна | К.э.н. | | |

По разделу «Социальная ответственность»

| Должность | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ООД ШБИП | Сечин Андрей Александрович | К.т.н. | | |

По разделу «Иностранный язык»

| Должность | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ОИЯ | Коротченко Татьяна Валериевна | К.ф.н. | | |

## ДОПУСТИТЬ К ЗАЩИТЕ:

| Руководитель ООП | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ОИТ | Шерстнев Владислав Станиславович | К.т.н. | | |

Томск – 2021 г.

**РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (КОМПЕТЕНЦИИ ВЫПУСКНИКОВ)**

по направлению 09.03.02 «Информационные системы и технологии»

| Код компетенции | Результат обучения (выпускник должен быть готов) |
|---|---|
| **Универсальные компетенции** | |
| **УК(У)-1** | Осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий. |
| **УК(У)-2** | Управлять проектом на всех этапах его жизненного цикла. |
| **УК(У)-3** | Организовывать и руководить работой команды, вырабатывая стратегию для достижения поставленной цели. |
| **УК(У)-4** | Применять современные коммуникативные технологии, в том числе на иностранном(ых) языке(ах), для академического и профессионального взаимодействия. |
| **УК(У)-5** | Анализировать и учитывать разнообразие культур в процессе межкультурного взаимодействия. |
| **УК(У)-6** | Определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки. |
| **Общепрофессиональные компетенции** | |
| **ОПК(У)-1** | Самостоятельно приобретать, развивать и применять математические, естественно-научные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте. |
| **ОПК(У)-2** | Разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач. |
| **ОПК(У)-3** | Анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями. |
| **ОПК(У)-4** | Применять на практике новые научные принципы и методы исследований. |
| **ОПК(У)-5** | Разрабатывать и модернизировать программное и аппаратное обеспечение информационных систем. |
| **ОПК(У)-6** | Использовать методы и средства системной инженерии в области получения, передачи, хранения, переработки и представления информации посредством информационных технологий. |
| **ОПК(У)-7** | Разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений. |
| **ОПК(У)-8** | Осуществлять эффективное управление разработкой программных средств и проектов. |
| **Профессиональные компетенции** | |
| **ПК(У)-1** | Управлять программно-техническими, технологическими и человеческими ресурсами. |
| **ПК(У)-2** | Управлять развитием баз данных. |
| **ПК(У)-3** | Управлять работами по сопровождению и проектами создания (модификации) информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы. |
| **ПК(У)-4** | Проектировать и организовывать учебный процесс по образовательным программам с использованием современных образовательных технологий. |
| **ПК(У)-5** | Осуществлять руководство разработкой комплексных проектов на всех стадиях и этапах выполнения работ. |

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа информационных технологий и робототехники (ИШИТР)
Направление подготовки 09.03.02 Информационные системы и технологии
Отделение школы (НОЦ) информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП

_____ _____ _____
(Подпись) (Дата) (Ф.И.О.)

**ЗАДАНИЕ**
**на выполнение выпускной квалификационной работы**

В форме:

| магистерской работы |
|---|

Студенту:

| **Группа** | **ФИО** |
|---|---|
| 8ИМ92 | Смирнову Павлу Олеговичу |

Тема работы:

| **Разработка сервиса для проверки зависимостей сборок проектов с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps** | |
|---|---|
| Утверждена приказом директора (дата, номер) | № 46-5/с от 15.02.2021 |

| Срок сдачи студентом выполненной работы: | 06.06.2021 |
|---|---|

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ:**

| | |
|---|---|
| **Исходные данные к работе**<br><br>*(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).* | Исходными данными для работы является ТЗ компании ООО «ITechx» по отслеживанию зависимостей сборок проектов с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps. А также статьи и материалы существующих методов и алгоритмов. Аппаратное и программное обеспечение заказчиков системы. |
| **Перечень подлежащих исследованию, проектированию и разработке вопросов**<br>*(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования;* | • Постановка цели и задачи исследования<br>• Анализ предметной области<br>• Анализ аналогов<br>• Общие требования к системе<br>• Проектирование архитектуры ПО<br>• Проектирование моделей БД<br>• Программная реализация сервиса |

| | |
|---|---|
| *обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).* | • Финансовый менеджмент, ресурсоэффективность и ресурсосбережение<br>• Социальная ответственность |
| **Перечень графического материала** | • Мультимедийная презентация |

**Консультанты по разделам выпускной квалификационной работы**

| Раздел | Консультант |
|---|---|
| «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение» | Верховская Марина Витальевна |
| «Социальная ответственность» | Сечин Андрей Александрович |
| «Иностранный язык» | Коротченко Татьяна Валериевна |

| | |
|---|---|
| **Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику** | **01.03.2021** |

**Задание выдал руководитель:**

| Должность | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ОИТ | Мирошниченко Евгений Александрович | К.т.н. | | |

**Задание принял к исполнению студент:**

| Группа | ФИО | Подпись | Дата |
|---|---|---|---|
| 8ИМ92 | Смирнов Павел Олегович | | |

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа информационных технологий и робототехники (ИШИТР)
Направление подготовки 09.03.02 Информационные системы и технологии
Отделение школы (НОЦ) информационных технологий

Форма представления работы:

| Магистерская работа |
|---|

## КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
## выполнения выпускной квалификационной работы

| Срок сдачи студентом выполненной работы: | 06.06.2021 |
|---|---|

| Дата контроля | Название раздела (модуля) / вид работы (исследования) | Максимальный балл раздела (модуля) |
|---|---|---|
| 14.01.21 – 26.02.21 | Анализ предметной области | 20 |
| 27.02.21 – 19.03.21 | Проектирование сервиса. | 20 |
| 20.03.21 – 28.04.21 | Реализация сервиса. | 20 |
| 29.04.21 – 14.05.21 | Финансовый менеджмент, ресурсоэффективность и ресурсосбережение | 15 |
| 15.05.21 – 22.05.21 | Социальная ответственность | 15 |
| 23.05.21 – 31.05.21 | Написание раздела на иностранном языке | 10 |

**СОСТАВИЛ:**
**Руководитель ВКР**

| Должность | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ОИТ | Мирошниченко Евгений Александрович | К.т.н. | | |

**СОГЛАСОВАНО:**

| Руководитель ООП | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ОИТ | Шерстнев Владислав Станиславович | К.т.н. | | |

# ЗАДАНИЕ ДЛЯ РАЗДЕЛА «ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

| Школа | **ИШИТР** | **Отделение школы (НОЦ)** | ОИТ |
|---|---|---|---|
| **Уровень образования** | Магистратура | **Направление/специальность** | 09.04.02 Информационные системы и технологии |

| **Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:** | |
|---|---|
| *1. Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих* | Материальные затраты: 299 руб.;<br>Амортизационные отчисления: 11 997 руб.;<br>Основная заработная плата на участников проекта: 341 348 руб.<br>Оклад инженера – 21760 руб. |
| *2. Нормы и нормативы расходования ресурсов* | - |
| *3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования* | Коэффициент отчислений во внебюджетные фонды – 30 %. |
| **Перечень вопросов, подлежащих исследованию, проектированию и разработке:** | |
| *1. Расчет инновационного потенциала НТИ* | SWOT-анализ;<br>технология QuaD;<br>Оценка научного уровня исследования. |
| *2. Расчет сметы затрат на выполнение проекта* | Расчет материальных затрат;<br>Расчет основной и дополнительной заработной платы;<br>Расчет отчислений во внебюджетные фонды;<br>Расчет бюджета проекта. |
| **Перечень графического материала** (с точным указанием обязательных чертежей)**:** | |
| *1. Технология QuaD*<br>*2. Матрица SWOT*<br>*3. Диаграмма Ганта;*<br>*4. График проведения НТИ* | |

| **Дата выдачи задания для раздела по линейному графику** | 01.03.2021 |
|---|---|

**Задание выдал консультант:**

| **Должность** | **ФИО** | **Ученая степень, звание** | **Подпись** | **Дата** |
|---|---|---|---|---|
| Доцент ОСГН ШБИП | Верховская Марина Витальевна | К.э.н. | | |

**Задание принял к исполнению студент:**

| **Группа** | **ФИО** | **Подпись** | **Дата** |
|---|---|---|---|
| 8ИМ92 | Смирнов Павел Олегович | | |

# ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

| Группа | ФИО |
|---|---|
| 8ИМ92 | Смирнов Павел Олегович |

| | | | |
|---|---|---|---|
| **Школа** | Инженерная школа информационных технологий и робототехники | **Отделение (НОЦ)** | Отделение информационных технологий |
| **Уровень образования** | Магистратура | **Направление/специальность** | 09.04.02 «Информационные системы и технологии» |

Тема ВКР:

| | |
|---|---|
| Разработка сервиса для проверки зависимостей сборок проектов с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps | |
| **Исходные данные к разделу «Социальная ответственность»:** | |
| 1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения | Рабочая зона оборудована местом, которое включает в себя: стул, компьютер с периферийными устройствами, расположенном на столе. Технологический процесс представляет собой работу с языком программирования C#, в интегрированной среде разработки Visual Studio. |
| Перечень вопросов, подлежащих исследованию, проектированию и разработке: | |
| **1. Правовые и организационные вопросы обеспечения безопасности:**<br>– специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства;<br>– организационные мероприятия при компоновке рабочей зоны. | Основные проводимые правовые и организационные мероприятия по обеспечению безопасности трудящихся в аудиториях. Анализ правильного расположения и организации рабочего места, а также режима работы. |
| **2. Производственная безопасность:**<br>2.1. Анализ выявленных вредных и опасных факторов<br>2.2. Обоснование мероприятий по снижению воздействия | – Отклонение показателей микроклимата;<br>– Недостаточная освещенность рабочей зоны;<br>– Превышение уровня шума;<br>– Опасные и вредные производственные факторы, связанные с электромагнитными полями;<br>– Психофизиологические факторы;<br>– Повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека; |
| **3. Экологическая безопасность:** | – Анализ воздействия объекта на литосферу: утилизация отходов, связанные с выходом из строя ПК, люминесцентных ламп и др. |
| **4. Безопасность в чрезвычайных ситуациях:** | Типичная ЧС – пожар. |

| | |
|---|---|
| | – разработка превентивных мер по предупреждению ЧС;<br>– разработка действий в результате возникшей ЧС и мер по ликвидации её последствий. |

| | |
|---|---|
| **Дата выдачи задания для раздела по линейному графику** | 01.03.2021 |

**Задание выдал консультант:**

| Должность | ФИО | Ученая степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ООД ШБИП | Сечин Андрей Александрович | К.т.н. | | |

**Задание принял к исполнению студент:**

| Группа | ФИО | Подпись | Дата |
|---|---|---|---|
| 8ИМ92 | Смирнов Павел Олегович | | |

## РЕФЕРАТ

Выпускная квалификационная работа содержит 104 страницы, 10 рисунков, 30 таблиц, 36 источников, 3 приложения.

Ключевые слова: разработка, определение зависимостей, сборка, триггер, конвейер обработки, облачные технологии, актуальность версий.

Объектом исследования является процесс определения зависимости сборок библиотек/модулей на актуальность версий.

Цель работы – разработка сервиса для проверки зависимостей сборок проектов на актуальность версий используемых в них внешних библиотек/модулей с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps.

В процессе исследования проводились анализ предметной области, анализ полученной информации, анализ существующих программных решений для выявления зависимостей, анализ требований к разрабатываемой платформе, реализация сервиса.

В результате исследования был разработан сервис для проверки зависимостей сборок проектов на актуальность версий используемых в них внешних библиотек/модулей. Данная версия включает в себя весь базовый функционал для отслеживания зависимостей.

Основные конструктивные, технологические и технико-эксплуатационные характеристики: используется архитектура клиент-сервер. На клиентской части применяются технологии Angular Framework 11, HTML 5, CSS 3, Bootstrap. На серверной части ASP.NET CORE 5.0. СУБД MS SQL Server, инфраструктура Microsoft Azure в качестве облачных вычислений.

Степень внедрения: разработана, протестирована, используется в рабочем процессе на предприятии.

Область применения: сервис на данный момент используется внутри компании для отслеживания проверки зависимостей сборок проектов на актуальность версий используемых в них внешних библиотек/модулей.

## ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

**Валидация:** Доказательство того, что требования конкретного пользователя, продукта, услуги или системы удовлетворены.

**Аутентификация:** Проверка подлинности пользователя путём сравнения введённого им пароля (для указанного логина) с паролем, сохранённым в базе данных пользовательских логинов.

**Trigger:** Процедура, которая не вызывается непосредственно пользователем, а исполняется при наступлении определенного события.

**Front-end:** Клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.

**Back-end:** Программно-аппаратная часть сервиса.

**Build:** Подготовленный для использования информационный продукт, чаще всего представленный в виде двоичного кода, содержащий исполняемый код.

**Framework:** Программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта.

**Web-сервис:** Сетевая технология, обеспечивающая межпрограммное взаимодействие на основе веб-стандартов.

**ПО:** Программное обеспечение.

**ООП:** Объектно-ориентированное программирование.

**БД:** База данных.

**СУБД:** Система управления базами данных.

**ТЗ:** Техническое задание.

**IDE:** Integrated Development Environment.

**DevOps:** Development and Operations.

**MVC:** Model-View-Controller.

**HTML:** Hyper Text Markup Language.

**CSS:** Cascading Style Sheets.

**ИС**: Информационная система.

## Оглавление

**Введение**

Облачные вычисления и технологии являются сегодня одним из ведущих трендов мирового ИТ рынка. Их обсуждают буквально все компании и аналитики, каждый так или иначе планирует их использование с целью создания продуктов и сервисов на их основе. Некоторые лишь прощупывают почву, для того, чтобы не оказаться позади и не упустить очевидные преимущества, которые предоставляют “облака”.

Этот феномен можно объяснить достаточно просто – наконец-то появляется решение, позволяющее существенно сократить затраты на ИТ-услуги, по-новому взглянуть на весь процесс автоматизации деятельности компаний и создания программного обеспечения, отказаться от высоких входных инвестиций в инфраструктуру и ее последующего поддержания, а также решить проблемы быстрого развертывания приложений, выхода на новые рынки, расширения клиентской базы, количества заказчиков и т.п.

Облачный подход позволяет организовать динамическое предоставление услуг, когда пользователи могут производить оплату по факту и регулировать объем своих ресурсов в зависимости от реальных потребностей без долгосрочных обязательств.

Облачные вычисления обладают многими преимуществами по сравнению с традиционными решениями для построения инфраструктур предприятий, предложению сервисов и услуг и т.п. Среди таких преимуществ выделяются:

- гибкость;
- масштабируемость;
- оплата за фактически использованные ресурсы;
- высокая надежность и отказоустойчивость.

Но несмотря на уже явное преимущество облачных сервисов и их обширное количество, предоставляемое различными компаниями, среди них всё ещё достаточно открытых задач для разработки.

Одной из таких задач является проверка зависимостей какого-либо проекта на актуальность версий используемых в нём внешних библиотек/модулей. Несмотря на наличие уже существующих на рынке программных решений, было предпринято намерение реализовать собственный сервис, который в узкой степени удовлетворяет потребностям компании при работе в облачной среде.

Целью работы является раскрытие понятия облачных технологий, анализ продуктов на основе технологии, доступных как простым пользователям, так и бизнесу, и разработчикам. Также необходимо разработать сервис для проверки зависимостей сборок проектов на актуальность версий используемых в них внешних библиотек/модулей с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps.

Сервис должен позволяет отслеживать и с легкостью управлять зависимостями различных версий сборок клиентских библиотек внутри проекта.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить обзор доступных материалов по данной теме;
- определить основные преимущества облачных технологий;
- провести анализ ПО на основе облачных технологий;
- решить практическую задачу, состоящую в разработке сервиса на основе облачной платформы Azure DevOps.

В разделе 1 рассмотрена теоретическая составляющая облачных технологии, её основные категории и понятия, а также преимущества и недостатки.

Раздел 2 содержит описание предметной области, особенности уже существующих конкретных продуктов с аналогичным функционаломи произведено их сравнение с разрабатываемым сервисом.

В разделе 3 предложена архитектура сервиса, разработана концептуальная модель базы данных, а также описаны используемые сущности, с которыми ведется работа в ходе разработки сервиса.

Раздел 4 посвящен программной реализации сервиса. Обоснован выбор среды разработки и языков программирования. Создана архитектура программного обеспечения платформы, описаны ее детали. Написание шаблонов для развертывания инфраструктуры в среде Azure.

В разделе 5 рассматривается пример работы разработанного сервиса в инфраструктуре платформы Azure DevOps, где показаны основные моменты работы взаимодействия с сервисом.

В разделе 6 дана оценка потенциальных потребителей результатов разработки, проведен SWOT-анализ, анализ конкурентных решений. Показан план этапов работы, определена трудоёмкость и построен календарный график, сформирован бюджет затрат.

Раздел 7 содержит анализ действующих стандартов безопасности труда при создании программного обеспечения.

Также был произведен перевод один из разделов дипломной работы на иностранный язык.

# 1. ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ

Перед тем как приступить к разбору темы, связанной с отслеживанием зависимостей сборок проектов от внешних библиотек/модулей, используемых в них, необходимо разобраться в какой среде и с какими сервисами среде ведётся работа.

В ходе работы все используемые ресурсы предоставляются облачной платформой и используют облачные вычисления.

Для более детального разбора темы облачных вычислений необходимо выяснить ответы на основные вопросы [1], которые помогут понять, что же это на самом деле:

- где располагаются приложения;
- основные характеристики облачных вычислений;
- предоставляемые сервисы;
- границы управляемости.

## 1.1 Где располагаются приложения?

Обсуждая облачные вычисления, следует обращать внимание на то, где располагаются приложения. В настоящее время существует три основных модели расположения приложений:

- в инфраструктуре заказчика;
- у компании-хостера;
- в облаке.

Расположение в инфраструктуре заказчика (on premises). Это наиболее традиционная модель развертывания приложений, существующая уже десятки лет. Размещение приложений в локальной инфраструктуре предполагает существенные начальные инвестиции в аппаратные ресурсы, программное обеспечение, сетевую инфраструктуру и персонал.

Такая модель – оплата, приобретение, владение – напрямую связана с высокими капитальными затратами, но, в тоже время, она обеспечивает полный контроль за инфраструктурой, аппаратным и программным обеспечением.

Расположение у компании-хостера (hosting). Такая модель развертывания приложений, называвшаяся ранее Application Services Prodiver (ASP), а затем – SaaS или просто «хостинг» получила свое развитие несколько лет назад и является одним из наиболее популярных способов снижения расходов на информационные технологии. Она основана на аренде аппаратной платформы, программного обеспечения, соответствующей инфраструктуры и персонала, выполняющего ее обслуживание. Такая модель отличается меньшим контролем за инфраструктурой, аппаратным и программным обеспечением и базируется на оплате фиксированного числа ресурсов, что обычно предполагает оплату даже в тех случаях, когда арендуемые ресурсы не используются.

Расположение в облаке (cloud). Данная модель появилась совсем недавно. Она предполагает оплату по факту использования арендуемых аппаратных и программных ресурсов, что приводит к существенному снижению начальных расходов и переходу от капитальных инвестиций к операционным расходам. Такая модель отличается практически отсутствием контроля за инфраструктурой и аппаратным обеспечением, а при аренде программного обеспечения – еще и отсутствием контроля за ним.

Каждый подход имеет свои достоинства и недостатки, но, с точки зрения экономики, самой важной характеристикой является оплата по факту использования, реализуемая именно облачными вычислениями. Таким образом [2]:

**Облачные вычисления** – это такой подход к размещению, предоставлению и потреблению приложений и компьютерных ресурсов, при котором приложения и ресурсы становятся доступны через Интернет в виде сервисов, потребляемых на различных платформах и устройствах. Оплата таких сервисов осуществляется по их фактическому использованию.

## 1.2 Основные характеристики облачных вычислений

В облачных вычислениях выделяют следующие ключевые характеристики [3]:

**Масштабируемость**

Масштабируемое приложение позволяет выдерживать большую нагрузку, за счет увеличения количества одно временно запущенных экземпляров. Как правило, для одновременного запуска множества экземпляров используется типовое оборудование, что снижает общую стоимость владения и упрощает сопровождение инфраструктуры.

**Эластичность**

Эластичность позволяет быстро нарастить мощность инфраструктуры, без необходимости проведения начальных инвестиций в оборудование и программное обеспечение. Эластичность связана с масштабируемостью приложений, так как решает задачу моментального изменения количества вычислительных ресурсов, выделяемых для работы информационной системы.

**Мультитенантность**

Мультитенантность – это один из способов снижения расходов за счет максимального использования общих ресурсов для обслуживания различных групп пользователей, разных организаций и категорий потребителей.

**Оплата за использование**

Оплата использованных ресурсов – это еще один атрибут облачных вычислений, позволяющий перевести часть капитальных издержек в операционные. Приобретая только необходимый объем ресурсов, можно оптимизировать расходы, связанные с работой информационных систем организации.

**Самообслуживание**

Самообслуживание позволяет потребителям запросить и получить требуемые ресурсы за считанные минуты.

Как можно заметить, только сочетание нескольких атрибутов облачных вычислений приводит к достижению задачи повышения доходов и снижения расходов. Так, оплата только использованных ресурсов максимально эффективна в сочетании с эластичностью инфраструктуры.

### 1.3 Облачные вычисления и предоставляемые ими сервисы

Облачные вычисления и предоставляемые ими сервисы (например, вычислительные мощности или хранилища) можно сравнить с коммунальными услугами. Так же как в жару или холод меняется потребление воды и электричества, так и потребление сервисов, предоставляемых «облачными» платформами, может возрастать или уменьшаться в зависимости от повышения или понижения нагрузок.

Схожесть сервисов и коммунальных услуг заключается в нескольких аспектах. Во-первых, и в том и в другом случае потребители платят только за реальную утилизацию. Во-вторых, и те, и другие ресурсы вы берете в аренду – т.е. в большинстве случаев вам не нужно подключаться к колодцу для получения воды или непосредственно к электростанции для получения электричества – поставщики таких сервисов обеспечивают их доступность в виде арендуемых «ресурсов», оставляя за собой вопросы создания и поддержания инфраструктуры. В-третьих, заключая договор с соответствующей организацией, вы подразумеваете доступность тех или иных ресурсов, а организация – своевременную оплату их аренды.

Какие сервисы чаще всего предоставляют «облачные» платформы? Хостинг приложений, хранение данных, проведение вычислений – вот наиболее частые сценарии использования «облачных» платформ. Говоря про «облачные» платформы и предоставляемые ими сервисы, обычно употребляют словосочетание «...как сервис». Можно выделить следующие основные сервисы, предоставляемые облачными платформами [4].

### 1.4 Облачные сервисы и границы управляемости

Обсуждая различные типы облачных сервисов – программное обеспечение, платформу и инфраструктуру как сервис, следует обращать

внимание на т.н. границы управляемости – т.е. на то, чем, в сравнении с традиционными моделями развертывания в собственной инфраструктуре, можно управлять при переходе на облачную платформу [5]. По понятным причинам, инфраструктура как сервис предоставляет большие возможности по настройке отдельных компонентов, тогда как платформа как сервис и программное обеспечение как сервис практически минимизируют эти возможности. Отличия в границах управляемости показаны на рисунке 1.

Из рисунка 1 видно, что при развертывании собственной инфраструктуры вы управляете всеми ее компонентами – от сетевых ресурсов до выполняющихся приложений. Тогда как при использовании модели IaaS вы можете контролировать такие компоненты, как среда исполнения кода, безопасность и интеграция, базы данных, и т.п.


Рисунок 1 – Границы управляемости

При переходе к модели PaaS, все компоненты платформы предоставляются как сервисы с ограниченными возможностями для управления ими. Это сделано, чтобы предоставить в распоряжение потребителей оптимально сконфигурированную платформу, не требующую дополнительных настроек.

## 1.5 Платформа Microsoft Azure

На рынке сегодня существует множество платформ для организации облачных вычислений.

Для того, чтобы выбрать наиболее подходящую платформу и провайдера необходимо четко сформулировать требования, предъявляемые к облаку, а также произвести пробное тестирование всех возможных платформ. Зачастую это наилучший способ понять, подходит ли решение или необходимо пробовать создавать свое на основе открытых платформ.

В ходе работы используется платформа Azure от компании Microsoft.

В 2010 году Microsoft начала предлагать компаниям сервисы ИТ инфраструктуры в виде веб-сервисов, то есть по модели, которая сегодня известна под названием «облачные вычисления».

Сегодня Azure предлагает в высшей степени надежную, масштабируемую и экономичную инфраструктурную платформу в облаке, которую используют в своей работе сотни тысяч компаний в 190 странах мира.

### 1.5.1 Обзор платформы

Это идеальная облачная платформа для Microsoft-ориентированных разработчиков и компаний. Впрочем, Windows Azure также поддерживает PHP, MySQL, Ruby on Rails, Python, Java, Eclipse и Zend. Главным преимуществом Azure перед Amazon Web Services и Rackspace Cloud является высокий уровень автоматизации, позволяющий разработчику думать только о своем приложении, а не об инфраструктуре. Кроме того, эта платформа позволяет легко интегрировать размещаемые на ней приложения с локальной ИТ инфраструктурой компании с помощью стандартов SOAP, REST и XML [6].

Платформа Azure предоставляет набор сервисов, которые, в основной массе, схожи с сервисами, используемыми разработчиками «традиционных» приложений:

– **Вычислительные сервисы.** Представляют собой контейнеры для приложений с поддержкой современных технологий разработки, включая .NET, Java, PHP, Python, Ruby on Rails и нативный код.

– **Сервисы хранения данных.** Масштабируемая распределенная система хранения данных, поддерживающая ряд моделей хранения, включая табличные структуры, бинарные объекты, асинхронные очереди сообщений, традиционные файловые системы и сети распределения контента (CDN, content distribution networks).

– **Коммуникационные сервисы.** Доступны через облачную сервисную шину и могут использоваться как средство обмена сообщениями или брокер соединений с другими облачными сервисами или сервисами, находящимися у заказчиков.

– **Сервисы обеспечения безопасности.** Сервисы управления доступом, основанные на политиках, которые поддерживают механизмы федерации и позволяют интегрироваться с существующими системами управления идентификацией.

– **Прикладные сервисы.** Компоненты и сервисы, которые могут использоваться для разработки облачных приложений и прикладных сервисов.

### 1.5.2 Компоненты облачной платформы

Azure состоит из множества облачных сервисов, которые можно использовать в комбинациях, соответствующих всевозможным бизнес-требованиям и организационным задачам (рисунок 2).

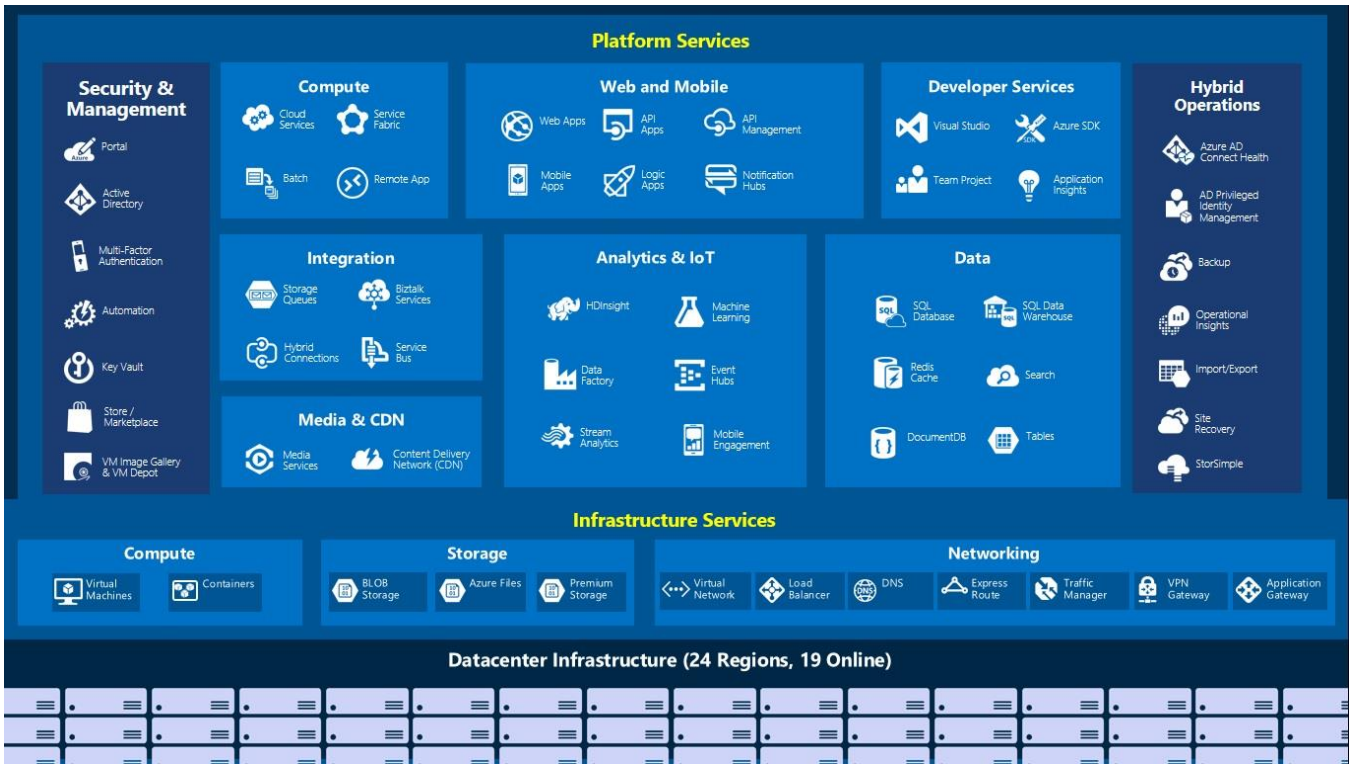



Рисунок 2 – Список облачных сервисов Azure

Далее представлены основные сервисы Azure по категориям, которые активно использовались в ходе работы [7].

### 1.5.2.1 Вычисления

- Сервис Virtual Machines позволяет пользователям запускать виртуальные машины общего назначения под управлением Microsoft Windows, Linux, а также преднастроенные образы популярных сервисных пакетов.
- Служба Приложений позволяет разработчикам публиковать веб-сайты и управлять ими.
- Сервис хостинга веб-сайтов позволяет разработчикам создавать сайты с использованием .NET, PHP, Node.js или Python и разворачивать их с использованием FTP, Git, Mercurial, Team Foundation Server или загружать через пользовательский портал.

### 1.5.2.2 Управление данными

- Сервисы хранения (Storage Services) предоставляют REST и SDK API для хранения данных в облаке и доступа к ним.

- SQL Database, предназначена для создания, масштабирования и расширения приложений в облаке с использованием технологии Microsoft SQL Server

#### 1.5.2.3 Работа в сети

- Load Balancer позволяет масштабировать приложения и обеспечивать доступность служб. Также поддерживает входящие и исходящие сценарии, обеспечивает низкую задержку и высокую пропускную способность.
- Шлюз приложений – это подсистема балансировки нагрузки веб-трафика, предназначенная для управления трафиком веб-приложений.

## 1.6 Azure DevOps

Внедрение облачных технологий в корне изменило способы создания, развертывания и эксплуатации приложений. Внедряя методологию DevOps, команды получают дополнительные возможности для улучшения методик работы и качества обслуживания клиентов.

Azure DevOps предоставляет сервисы для разработчиков для поддержки групп по планированию работы, совместной работе над разработкой кода, а также по созданию и развертыванию приложений [8].

### 1.6.1 DevOps методология

DevOps — методология активного взаимодействия специалистов по разработке со специалистами по информационно-технологическому обслуживанию, и взаимная интеграция их рабочих процессов друг в друга для обеспечения качества продукта [9].

DevOps влияет на жизненный цикл приложения на всех этапах — от планирования и разработки до доставки и эксплуатации. Каждый этап зависит от других, но сами этапы не зависят от ролей, выполняемых сотрудниками компании. В полноценно реализованной культуре DevOps каждая роль в той или иной степени задействована на каждом этапе.

Помимо внедрения культуры DevOps, команды воплощают в жизнь подход DevOps, применяя определенные методики на протяжении всего

жизненного цикла приложения. Некоторые из этих методик помогают ускорить, автоматизировать и улучшить выполнение конкретного этапа. Другие охватывают несколько этапов, помогая командам создавать целостные процессы, которые помогают повысить продуктивность. Важно понимать, что с применением методологии DevOps данные методики по большей части автоматизированы, за счёт чего достигается высокая производительность в команде, т.к. пропускаются рутинные шаги, постоянно возникающие в процессе разработки.

Далее перечислены основные методики применяемые при данной методологии:

#### 1.6.1.1 Непрерывная интеграция (Continuous Integration)

Это практика разработки программного обеспечения, которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.

#### 1.6.1.2 Автоматическое тестирование (Automated Testing)

Это процесс верификации программного обеспечения, при котором основные функции и шаги теста, такие как запуск, инициализация, выполнение, анализ и выдача результата, выполняются автоматически при помощи инструментов для автоматизированного тестирования.

#### 1.6.1.3 Непрерывное развертывание (Continuous Deployment)

Объединение Continuous Integration (непрерывной интеграции) и Continuous Delivery (непрерывной поставки). Это следующий шаг после успешной сборки и успешного прохождения автоматических тестов (и, возможно, установки галочки «сборка готова к развертыванию» ответственным человеком). Если вы уверены в ваших тестах, их наборе и покрытии, при их успешном выполнении запускается автоматическая установка на соответствующую среду, тестовую или продуктовую.

#### 1.6.1.4 Инфраструктура как код (Infrastructure as Code)

Это процесс управления и подготовки вычислительной инфраструктуры (процессы, физические сервера, виртуальные сервера и т.п.) и их конфигурации через машинно-обрабатываемые файлы определений, а не физическую конфигурацию оборудования или использование конфигурационных инструментов.

#### 1.6.1.5 Мониторинг быстродействия приложения (Application Performance Monitoring)

Это мониторинг и управление быстродействием и доступностью ПО. APM стремится выявлять и диагностировать проблемы быстродействия комплексно, для поддержания ожидаемого уровня обслуживания.

#### 1.6.1.6 Нагрузочное тестирование (Load Testing)

Подвид тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству).

#### 1.6.1.7 Управление релизами (Release Management)

Управление релизом заключается в том, что мы определяем формальные критерии, готова ли сборка к установке на соответствующую среду. Примером критериев, по которым сборка готова, и, если они выполняются, автоматически запускается поставка (Continuous Deployment) могут быть:

- DEV среда – сборка прошла без ошибок.
- STAGE среда – сборка была установлена на DEV среде и unit тесты прошли успешно.
- PROD среда – сборка прошла тестирование на STAGE среде, есть не более 5% minor ошибок, major ошибок нет.

### 1.6.2 Pipelines (конвейеры)

Для обеспечения максимальной эффективности команды разработчиков и ускорения выхода на рынок важно спланировать и реализовать развертывание конвейеров DevOps CI и CD.

Конвейер CI/CD описывает подход, который упрощает процесс объединения вновь написанного кода с существующим кодом. Эта концепция также позволяет запускать различные типы тестов на каждом этапе и завершать тестирование, запуская и развертывая вновь написанный код в уже протестированной версии программного обеспечения, которую видят конечные пользователи.

**Конвейер DevOps CI/CD** - необходимый атрибут разработки программного обеспечения, который включает любой из подходов Agile. Именно здесь стоит вопрос автоматизации тестирования, которая обеспечивает максимально быструю обратную связь между отдельными командами специалистов, работающих над проектом [10].

Что касается основных этапов конвейерного процесса DevOps, их всего пять:

1. настройка инфраструктуры (фреймворка) DevOps конвейера CD/CI;
2. интеграция с инструментом управления исходным кодом;
3. подключение создающего инструмента автоматизации (building automation tool);
4. развертывание программного обеспечения на стороне сервера;
5. тестирование программного кода.

Исходя из вышесказанного, конечные преимущества развертывания DevOps конвейера включают в себя:

- более быстрый жизненный цикл разработки программного обеспечения;
- внедрение эффективных решений по обеспечению качества;
- автоматизация создания и выпуска программного обеспечения;

- настройка плавной работы программного обеспечения.

Фактически, конвейер описывает набор методов для автоматизации развертывания приложений в различных средах, который позволяет вам делать выпуски более частыми, уменьшать количество трудно исправляемых ошибок и связанных с ними простоев, а также ускорять работу различных групп.

# 2. ПРОВЕРКА ЗАВИСИМОСТЕЙ СБОРОК ПРОЕКТА

В реальных проектах в большинстве случаев используется многочисленное количество дополнительных библиотек/модулей. А зачастую также бывает, что один клиентский проект, может зависеть от другого.

В ходе разработки различных проектов в разные библиотеки/модули вносится новая функциональность и создаются новые версии продукта. Как же отслеживать все данные зависимости с различными версиями, да ещё когда при разработке применяется подход с использованием конвейеров непрерывной интеграции (CI) и непрерывного развертывания (CD), чтобы не было простоя между выпуском новых версий какого-либо продукта?

## 2.1 Сценарий проверки зависимостей

В ходе работы используются конвейеры в Azure DevOps для CI/CD. Когда запрос на добавление новой функциональности завершен, и включенный код добавляется в основную ветку репозитория, сборка запускается автоматически. Это основная функциональность любого конвейера, автоматизации развертывания приложений в различных средах. Если изменение кода происходит в библиотеке, Azure DevOps предоставляет только возможность построить именно эту библиотеку, в которую были внесены изменения. Проекты, зависимые от данной библиотеки и также использующие её, не могут быть построены автоматически. Например, изменения не могут быть немедленно протестированы. Сборки пользователей должны быть запущены вручную. Данный факт идёт в разрез с основными принципами DevOps методологии об автоматизации рабочих процессов.

Чтобы обойти эту проблему, необходима дополнительная служба проверки зависимостей, основанная на триггерах конвейеров. Подразумевается, что данная служба информирует об успешно завершенных сборках, а затем, при необходимости запускает другие зависимые конвейеры.

## 2.2 Azure WebHooks

Запуск конвейеров в среде Azure DevOps можно выполнить двумя способами: вручную или с помощью триггера. Выполнение конвейера вручную также называется выполнением по требованию, т.е. пользователь самостоятельно вызывает запуск конвейера, соответственно данный способ мало пригоден для методологии DevOps, т.к. требуется постоянный контроль и определенный набор действий со стороны пользователя.

Данную проблему прекрасно решает другой способ запуска конвейера, применение триггеров в ходе работы. Триггеры обозначает единицу обработки, которая определяет время запуска для выполнения конвейера. Несколько триггеров могут запускать один конвейер, или один триггер может запускать несколько конвейеров. И всё это может быть автоматизировано благодаря написанию сценариев для триггера.

Сейчас в среде Azure DevOps поддерживается три типа триггеров [11]:

- Триггер планировщика. Триггер, который запускает конвейер в определенное время по расписанию;
- Триггер "переворачивающегося" окна. Триггер, который работает на основе периодических интервалов, сохраняя состояние;
- Триггер на основе событий: триггер, который реагирует на событие.

Благодаря данному способу возможно с лёгкостью отслеживать происходящие события в проектах Azure DevOps, что предоставляет доступ к новой функциональности: Azure WebHooks (сервисный крючок), иными словами служебные перехватчики.

Azure WebHooks позволяют запускать задачи в других службах. Служебные перехватчики также могут быть использованы в настраиваемых приложениях и службах как более эффективный способ управлять действиями, когда в ваших проектах происходят события.

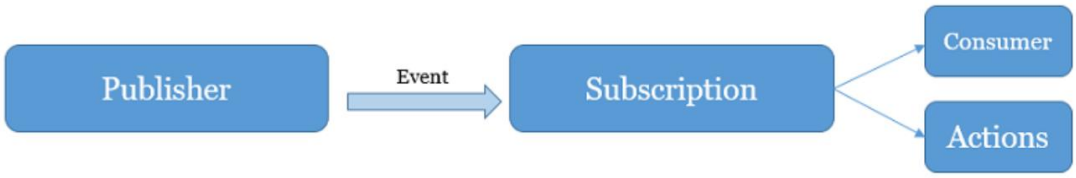



Рисунок 2 – Логика работы Azure WebHooks

Azure WebHooks основаны на вполне понятной логике событийной модели. Publisher (издатели) сервисных ловушек определяют набор событий. Subscription (подписки) прослушивают события и определяют Action (действия), которые необходимо предпринять в зависимости от события. Подписки также нацелены на Consumer (потребителей), которые представляют собой внешние службы, которые могут выполнять свои собственные действия при возникновении события.

### 2.3 Анализ существующих программных решений

В данном разделе рассматриваются уже существующие программные продукты с точки зрения возможности отслеживания зависимостей проекта на актуальность версий используемых в нём внешних библиотек/модулей.

#### 2.3.1 Dependabot

Действительно, при разработке время от времени приходится сталкиваться с тем, что та или иная библиотека обновляется, либо (реже, но всё же) выходит из употребления и помечается как непроверенная. В последнем случае возникает необходимость искать равноценную замену этому модулю - но даже если библиотека не вышла из эксплуатации, а просто обносилась, имеет смысл использовать актуальную её версию. И вот здесь нам поможет система Dependabot.

Dependabot — это приложение GitHub, которое автоматически устанавливается в каждом репозитории, в котором включены автоматические обновления безопасности, и создает запросы на извлечение для обеспечения безопасности и актуальности зависимостей.

Каждый день Dependabot проверяет ваши файлы зависимостей на наличие устаревших требований и открывает индивидуальные запросы на получение всех найденных файлов. Вы просматриваете PR, объединяете их и начинаете работать над последними, наиболее безопасными выпусками.

До этого Dependabot был автономным платным сервисом, пока он не был приобретен GitHub и напрямую интегрирован в GitHub, что сделало его бесплатным.

### 2.3.2 WhiteSource Renovate

Данная платформа также позволяет снижать риски безопасности благодаря тому, что подталкивает разработчиков к поддержанию зависимостей в актуальном состоянии.

WhiteSource Renovate регулярно проверяет зависимости проекта на предмет их устаревания и пытается помочь разработчику, автоматически открывая PR.

Платформа распространяется в качестве Open Source проекта на полностью бесплатной основе.

Для того чтобы начать использовать WhiteSource Renovate, нужно иметь репозиторий GitHub, где, в свою очередь, должен содержаться код проекта, зависимости которого нужно проверять. Принцип работы прост: указывается репозиторий, который нужно проверять, и WhiteSource Renovate анализирует его на предмет наличия небезопасных и устаревших зависимостей. Если таковые находятся, WhiteSource Renovate создаёт запрос на притягивание изменений с объявлением новых зависимостей. Далее остаётся запустить тесты, проанализировать результат - и можно применять изменения.

WhiteSource Renovate поддерживает целый ряд языков и языковых платформ (некоторые пока в виде альфа- и бета-версий): Ruby, JavaScript, Python, PHP, Rust, Go, .NET. Имеется поддержка Docker и Terraform.

### 2.3.3 Результаты анализа

Из проведенного анализа возможностей программных продуктов в области отслеживания зависимостей программного проекта на актуальность версий используемых в нём внешних библиотек/модулей видно, что почти все из них обладают возможностью в той или иной мере проверять файлы зависимостей проекта на наличие устаревших требований. Стандартным набором является создание запросов на извлечение для обеспечения безопасности и актуальности зависимостей.

Таблица 1 – Сравнение аналогов

| **Наименование продукта** | **Достоинства** | **Недостатки** |
|---|---|---|
| Dependabot [12] | • Поддержка большого количества языков программирования;<br>• Высокая надёжность;<br>• Лёгок в настройке, т.к. автоматически встроен в каждый репозиторий на GitHub;<br>• Расширенная документация;<br>• Бесплатен с ограниченным функционалом; | • Совместим лишь с веб-сервисом для хостинга GitHub;<br>• Закрытый исходный код;<br>• Нестабильная работа с конвейерами обработки (pipelines); |
| WhiteSource Renovate [13] | • Поддержка большого количества языков программирования;<br>• Открытый исходный код;<br>• Расширенная документация;<br>• Бесплатен; | • Совместим лишь с веб-сервисом для хостинга GitHub;<br>• Нестабильная работа с конвейерами обработки (pipelines); |
| Разрабатываемый сервис | • Возможность гибкой настройки под требования компании;<br>• Совместимость с используемой в компании платформой Azure DevOps;<br>• Изначальная разработка под функциональность конвейеров обработки (pipelines);<br>• Бесплатен; | • Наличие багов на старте, т.к. сервис пишется с нуля;<br>• Умение написания конфигурационного файла для перечисления зависимостей; |

Многие аналоги отлично справляются с функциональностью проверки файлов зависимостей на наличие устаревших требований и открытием индивидуальных запросов на получение всех найденных файлов.

Также аналоги имеют уже написанную и детально описывающую документацию по настройке, но также среди них имеется пара существенных недостатков из-за которых возникла идея создать собственный сервис для отслеживания зависимостей.

Вся работа для поддержки групп по планированию работы, совместной работе над разработкой кода, а также по созданию и развертыванию приложений в компании ведется в платформе Azure DevOps, а код хранится в веб-хостинге Azure Repos. Аналоги же в свою очередь жёстко привязаны к веб-хостингу GitHub, что приводит к невозможности их применения в используемой среде.

Также важным аспектом является, что вся работа по автоматизации развёртывания кода в компании ведётся с применением конвейеров обработки (pipelines), а аналогичные программные продукты не имеют гибкой настройки при работе с данными средствами. Отсюда вытекает следующий пункт за разработку собственного сервиса: гибкость в настройке под нужды компании. Т.к. сервис пишется с нуля и имеется полный доступ к исходному коду программы, то возможно сконфигурировать сервис в любом желаемом виде.

Исходя из результатов анализа, можно сделать вывод о том, что нет сервиса для отслеживания зависимостей, который разом бы решал все необходимые задачи, а те единицы что существуют, очень сильно привязаны к используемым средам и не слишком гибки в настройке. Следовательно, актуально создание такого сервиса.

# 3. ПРОЕКТИРОВАНИЕ СЕРВИСА

На этапе проектирования требования, изложенные в техническом задании, преобразуются в конкретные требования к внутреннему устройству и функционированию будущей платформы.

Были проработана архитектура будущего сервиса и структура базы данных.

## 3.1 Проектирование архитектуры

В основе работы приложения лежит бессерверный подход к реализации приложения [14].

Данный подход подразумевает не отсутствие серверов, а меньшую зависимость от них. Бессерверный код управляется событиями. Код может запускаться чем угодно — от традиционных веб-запросов HTTP до таймера или результата передачи файла. Инфраструктуру в бессерверной архитектуре можно моментально масштабировать в зависимости от потребностей, оплата за облачные вычисления происходит дискретно, только за то, что действительно используется.

Бессерверная архитектура еще меньше привязана к серверам и сосредоточена на коде, управляемом событиями. Разработчики занимаются не платформой, а отдельными микрослужбами с определенными функциями. У разработчиков бессерверного кода есть два главных вопроса:

- Что активирует код?
- Что делает код?

Для бессерверного кода инфраструктура не так важна. В некоторых случаях разработчик вообще больше не думает о размещении. Веб-запросами может управлять экземпляр IIS, Kestrel, Apache или другой веб-сервер, разработчик занимается только триггером HTTP. Триггер предоставляет стандартные кроссплатформенные полезные данные для запроса.

На следующем рисунке показаны четыре бессерверных компонента. HTTP-запрос активирует выполнение кода Pipeline Trigger API, который

выполняет поиск зависимостей. Pipeline Trigger API вставляет код в базу данных, после чего вызывает функцию для выполнения задачи запуска необходимых зависимых конвейеров.

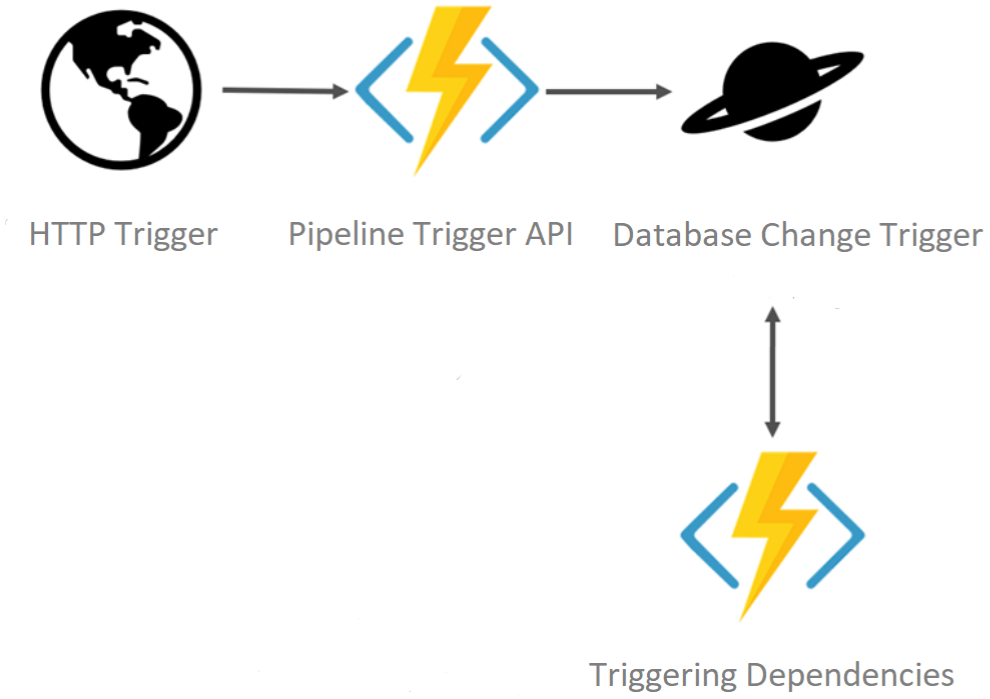



Рисунок 3 – Модель взаимодействия сервиса

Бессерверная архитектура обеспечивает четкое разделение кода и среды его размещения. Код реализуется в функции, которая вызывается триггером. После выхода из этой функции все необходимые ресурсы можно освободить. Триггером может быть выполняемая вручную операция, процесс с заданным временем выполнения, HTTP-запрос или передача файла. Результатом триггера является выполнение кода.

### 3.2 Описание структуры базы данных

К моменту преступления к программной реализации данного сервиса была разработана логическая модель базы данных.

На рисунке 4 представлены сущности БД, с которыми ведется работа.

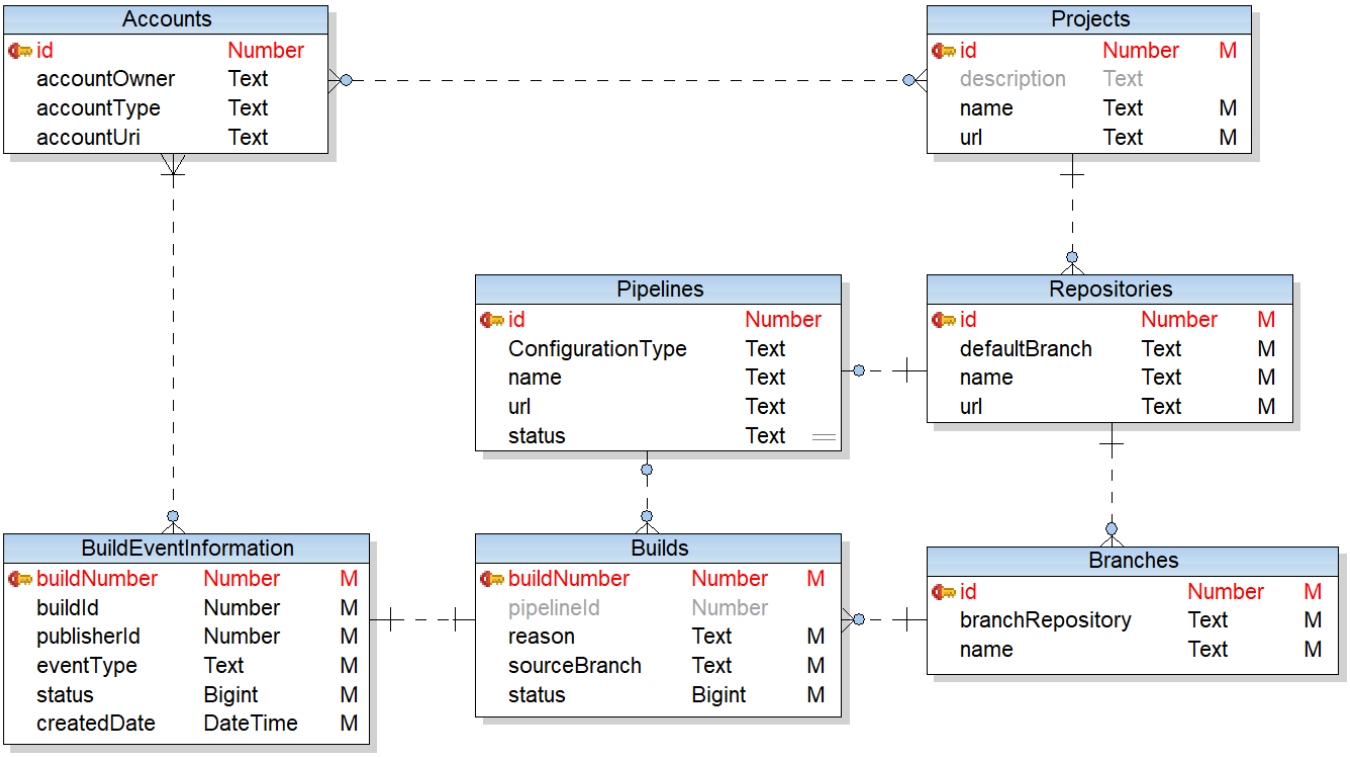



Рисунок 4 – Логическая модель БД

Более детальное описание некоторых сущностей:

**Account** – сущность описывающая аккаунт пользователя.

Таблица 2 – Сущность аккаунт

| Идентификатор | Тип | Может быть пустым | Комментарий |
|---|---|---|---|
| accountId | integer | No | Идентификатор учетной записи |
| accountOwner | string | | Владелец счета |
| accountType | string | No | Тип аккаунта: Личный, Корпоративный |
| accountUri | string | No | Uri учетной записи |

**Project** – сущность описывающая проект в среде Azure DevOps.

Таблица 3 – Сущность проекта

| Идентификатор | Тип | Может быть пустым | Комментарий |
|---|---|---|---|
| id | integer | No | Идентификатор проекта |
| description | string | Yes | Описание проекта |
| name | string | No | Имя проекта |
| url | string | No | URL-адрес проекта |

**Repositories** – сущность описывающая репозитории.

Таблица 4 – Сущность репозитория

| Идентификатор | Тип | Может быть пустым | Комментарий |
|---|---|---|---|
| id | integer | No | Идентификатор репозитория |
| defaultBranch | string | No | Имя ветви по умолчанию |
| name | string | No | Имя репозитория |
| url | string | No | URL-адрес репозитория |

**Branches** – сущность описывающая ветви репозитория.

Таблица 5 – Сущность ветви репозитория

| Идентификатор | Тип | Может быть пустым | Комментарий |
|---|---|---|---|
| id | integer | No | Идентификатор репозитория |
| branchRepository | string | No | Имя репозитория, которому принадлежит ветвь |
| name | string | No | Имя ветви |

**Pipelines** – сущность описывающая конвейеры обработки.

Таблица 6 – Сущность конвейера обработки

| Идентификатор | Тип | Может быть пустым | Комментарий |
|---|---|---|---|
| folder | string | No | Папка конвейера |
| id | integer | No | Идентификатор конвейера |
| name | string | No | Название конвейера |
| url | string | No | URL-адрес конвейера |
| configurationType | string | No | Тип конфигурации |

**Builds** – сущность описывающая сборку проекта.

Таблица 7 – Сущность сборки проекта

| Идентификатор | Тип | Может быть пустым | Комментарий |
|---|---|---|---|
| buildNumber | string | No | Номер сборки |
| pipelineId | integer | No | конвейер, связанный со сборкой |
| reason | string | No | Причина, по которой была создана сборка |
| sourceBranch | string | No | Исходная ветвь |
| status | string | No | Статус сборки |

**BuildEventInformation** – сущность описывающая событие, которое вызвало сборку.

Таблица 8 – Сущность событие триггера сборки

| Идентификатор | Тип | Может быть пустым | Комментарий |
|---|---|---|---|
| id | integer | No | Идентификатор события |
| buildId | integer | No | Идентификатор сборки |
| eventType | string | No | Тип события |
| publisherId | string | No | Идентификатор издателя |
| createdDate | date | No | Дата создания сборки |
| status | string | No | Статус сборки |

# 4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СЕРВИСА

## 4.1 Выбор среды разработки

При выборе языка программирования существует множество альтернатив: C++, C#, Java, Python и т. д. Такое же разнообразие наблюдается при выборе интегрированных сред разработки: Microsoft Visual Studio, Macromedia Studio, Eclipse, Borland Delphi, Sun Java Studio и другие.

Основным же критерием при выборе среды разработки является факт того что в техническом задании от предприятия прописан пункт реализации веб-приложения с помощью технологии ASP.NET Core 5.0.

Технология ASP.NET Core 5.0 представляет технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов [15].

Процесс реализации проекта проходил в среде разработки Visual Studio 2019 Professional. Данная среда включает в себя всё необходимое для максимизации производительности программиста: редактор исходного кода, встроенный отладчик, встроенный web-сервер, поддержка множества языков программирования и другие полезные средства.

## 4.2 Выбор языков программирования

При создания веб-приложения на используется платформа ASP.NET CORE 5.0, таким образом для написания бизнес логики на стороне сервера используется язык C#.

Немаловажным фактором является степень владения данным языком программирования и опыт работы в данной интегрированной среде разработки. В процессе обучения все работы выполнялись на языке программирования C#, что позволило применить полученные навыки при создании данного веб-приложения.

Скрипты для развертывания инфраструктуры посредствам кода были описаны в формате JSON. Скрипт для определения конвейеров обработки реализован в формате YAML.

### 4.3 Выбор СУБД

В ходе работы была использована система управления базами данных MS SQL Server, т.к. базы данных находится под управлением данной системы.

Данная СУБД соответствует всем стандартам SQL и входит в пакет разработки Visual Studio, а также имеет большое сообщество в сети.

Из явных преимуществ данной СУБД над другими можно выделить следующие пункты [16]:

- Масштабируемая – поэтому работать с ней можно на портативных ПК или мощной мультипроцессорной технике. Процессор может одновременно обрабатывать большой объем запросов;
- Высокая производительность – не первый год Microsoft подтверждает высокую производительность SQL Server транзакционными тестам и тестами производительности хранилищ данных;
- Большое сообщество - существует довольно большое сообщество в котором вы запросто найдёте ответы на свои вопросы;
- Расширения – существует возможность расширения функционала за счет сохранения своих процедур;
- Azure Data Studio – это упрощенное кроссплатформенное графическое средство управления и редактор кода. Позволяет создавать запросы к реляционным и нереляционным базам данных с поддержкой разных операционных систем и источников данных.

### 4.4 Развёртывание инфраструктуры посредствам кода

В ходе работы для создания инфраструктуры, следуя методологии DevOps, применялся подход Infrastructure as Code.

Чтобы реализовать инфраструктуру как код для решений Azure, используйте шаблоны Azure Resource Manager (шаблоны ARM). Шаблон —

это файл нотация объектов JavaScript (JSON), который определяет инфраструктуру и конфигурацию проекта. Шаблон использует декларативный синтаксис, который позволяет указать объект, который необходимо развернуть. При этом, для развертывания объекта, не нужно писать последовательность команд. В шаблоне указываются ресурсы для развертывания и свойства этих ресурсов [17].

Преимущества развертывания инфраструктуры, реализовывая подход Infrastructure as Code:

- Декларативный синтаксис;
- Повторяемые результаты;
- Оркестрация;
- Создание любого ресурса Azure;
- Встроенная проверка;
- Интеграция CI/CD;
- Экспортируемый код.

Также следуя данному подходу был написан шаблон для создания конвейера обработки и логика публикации кода приложения на сервере.

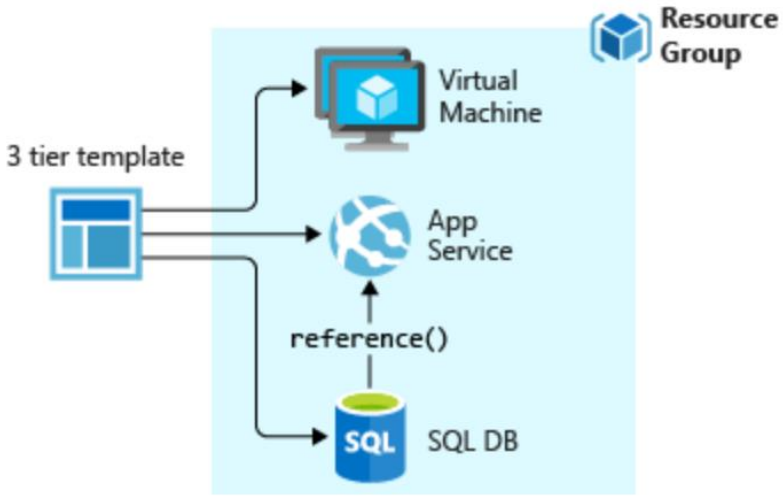



Рисунок 5 – Группа ресурсов, создаваемая благодаря связке данных шаблонов

Таким образом, благодаря связке шаблонов за считанные секунды было развёрнуто трехуровневое приложение в одной группе ресурсов.

## 5. ПРИМЕР РАБОТЫ РАЗРАБОТАННОГО СЕРВИСА

В данном разделе рассматривается пример работы разработанного сервиса в инфраструктуре платформы Azure DevOps, где показаны основные моменты работы взаимодействия с сервисом.

**Панель конвейеров обработки**

При загрузке платформы Azure DevOps и после прохождения процесса авторизации в системе, пользователю предоставляется доступ к панели отображения всех имеющихся конвейеров обработки, которые созданы в проекте компании (рисунок 6).

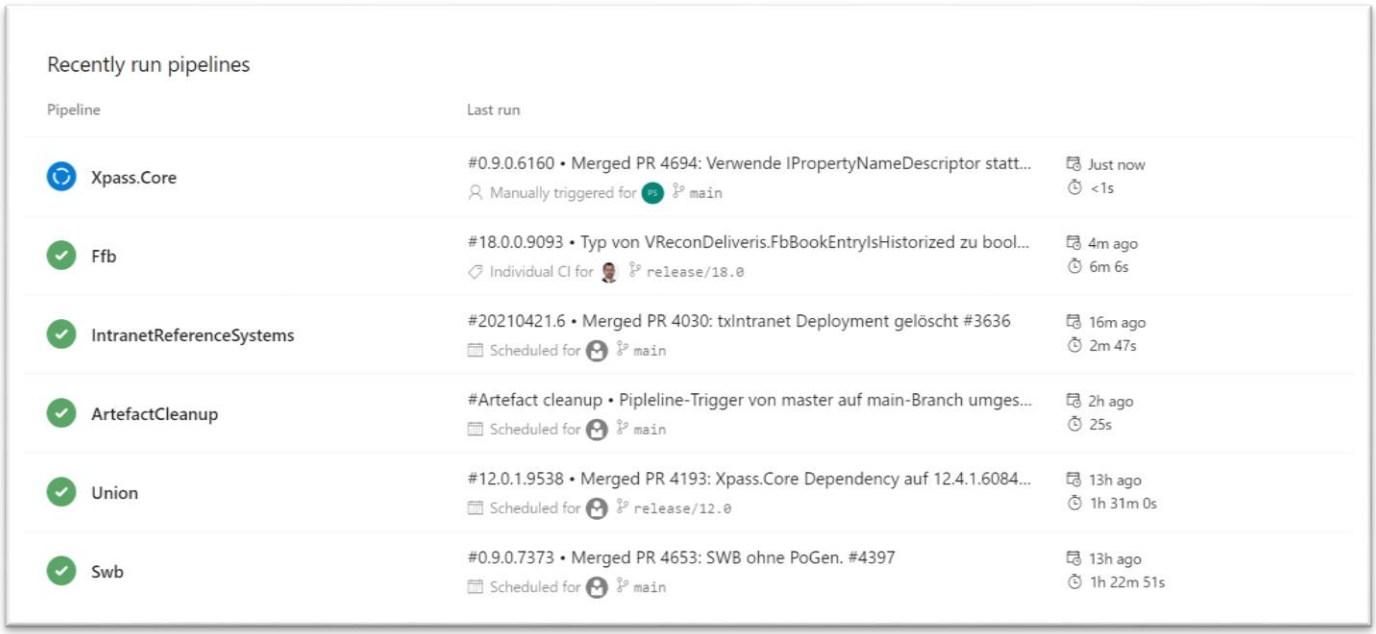



Рисунок 6 – Панель конвейеров обработки

На данной панели пользователь может видеть статус последнего конвейера обработки для каждого репозитория зарегистрированного в проекте компании.

Для каждого запуска предоставляется следующая информация по конвейеру обработки:

- Название;
- Текущий статус;
- Информация по последнему запуску;
- Причина;

- Время сборки;
- Пользователь запустивший сборку.

Исходя из данного изображения можно видеть, что все конвейеры обработки в состоянии завершенности, кроме конвейера “Xpass.Core”. Он находится в состоянии обработки.

**Типы запуска конвейеров обработки**

Конвейеры обработки в Azure DevOps могут иметь разные версии конвейера в разных ветвях, что может повлиять на то, какая версия триггеров конвейера будет оцениваться, и какая версия конвейера должна выполняться (рисунок 7).

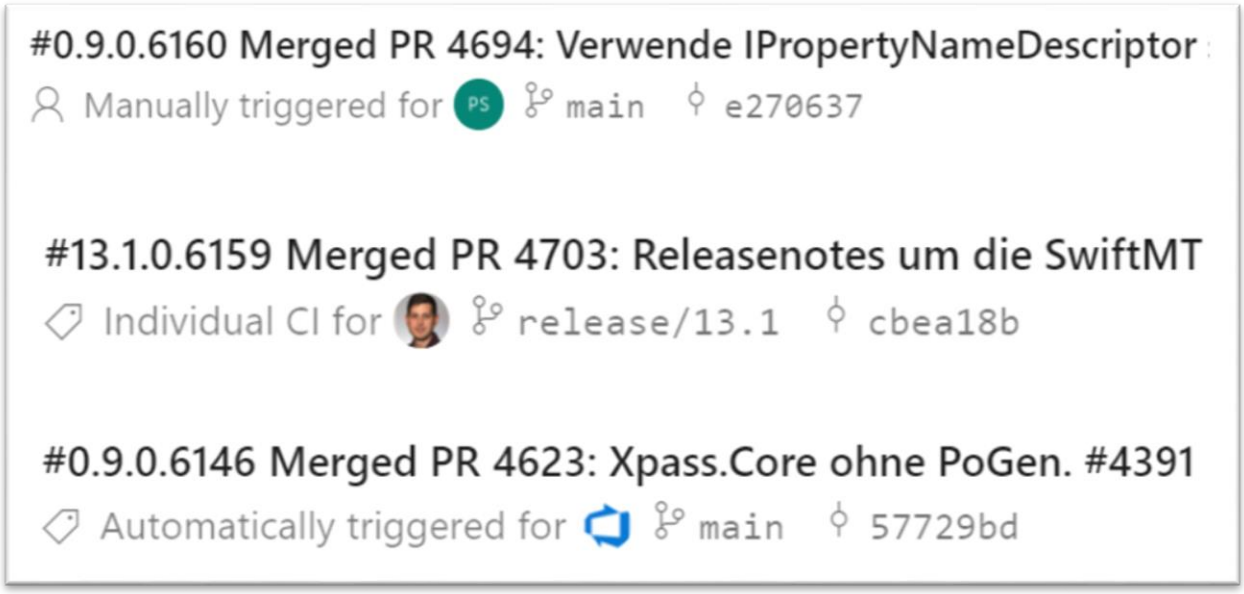



Рисунок 7 – Отображение различных типов запуски конвейеров обработки

Существуют следующие типы триггеров:

- Триггер Continuous Integration – используется версия конвейера в отправленной ветви;
- Триггер Pull Request – используется версия конвейера в исходной ветви для запроса на включение внесенных изменений;
- Запланированные триггеры – запланированные триггеры настраивают конвейер для выполнения по расписанию;

- Автоматический триггер – данный триггер используется при автоматическом запуске конвейера обработки с помощью портала Azure DevOps.

Автоматический тип триггера как раз соответствует причине автоматического запуска конвейера посредством зависимостей.

Иконка отображает не пользователя, запустившего сборку, а логотип Azure DevOps, что свидетельствует тому что причиной запуска послужил сам портал Azure DevOps посредствам созданного сервиса.

**Структура файла зависимостей**

Каждый репозиторий и ветвь репозитория содержит файл зависимостей “azure-pipelineDependencies.json” с индивидуальным содержимым. Файл представлен в формате json где содержится массив зависимостей репозиториев от которых непосредственно зависит данный конвейер обработки.

Так глянем пример данного файла зависимостей в репозитории “Dz” на ветке main (рисунок 9). Элемент массива содержит имя и ветку репозитория. Так, например, в данном списке мы можем видеть репозиторий “Xpass.Core” с указанной ветвью main. Данный факт свидетельствует о том, что существует зависимость между данными репозиториями (т.к. Dz зависит от Xpass.Core).

Если глянуть рисунок 8, то репозиторий с такими параметрами и был запущен.

azure-pipelineDependencies.json

Contents History Compare Blame

```
{
    "pipelineDependencies": [
        {
            "repositoryName": "Alert",
            "branchName": "main"
        },
        {
            "repositoryName": "Xpass.Core",
            "branchName": "main"
        },
        {
            "repositoryName": "SwiftMT",
            "branchName": "main"
        },
        {
            "repositoryName": "WixSetup",
            "branchName": "main"
        },
        {
            "repositoryName": "WpioDz",
            "branchName": "main"
        },
        {
            "repositoryName": "Quickfix.Wrapper",
            "branchName": "main"
        }
    ]
}
```

Рисунок 9 – Структура файла зависимостей

**Автоматический запуск зависимостей**

Так как между репозиториями “Dz” и “Xpass.Core” существует зависимость, и то что на рисунке 8 видим, что был запущен конвейер обработки репозитория “Xpass.Core”, то на рисунке 10 наблюдаем автоматическую сборку конвейра обработки репозитория “Dz”.

Также от конвейера обработки репозитория “Xpass.Core” зависят и другие к конвейеры, что мы и видим на рисунке 10. Автоматически одновременно запускаются несколько конвейеров обработки зависимых репозиториев. Некоторые из них уже закончили сборку, а некоторые еще в процессе.

Также после данных сборок, могут автоматически начаться другие зависимости в последовательности цепочки.

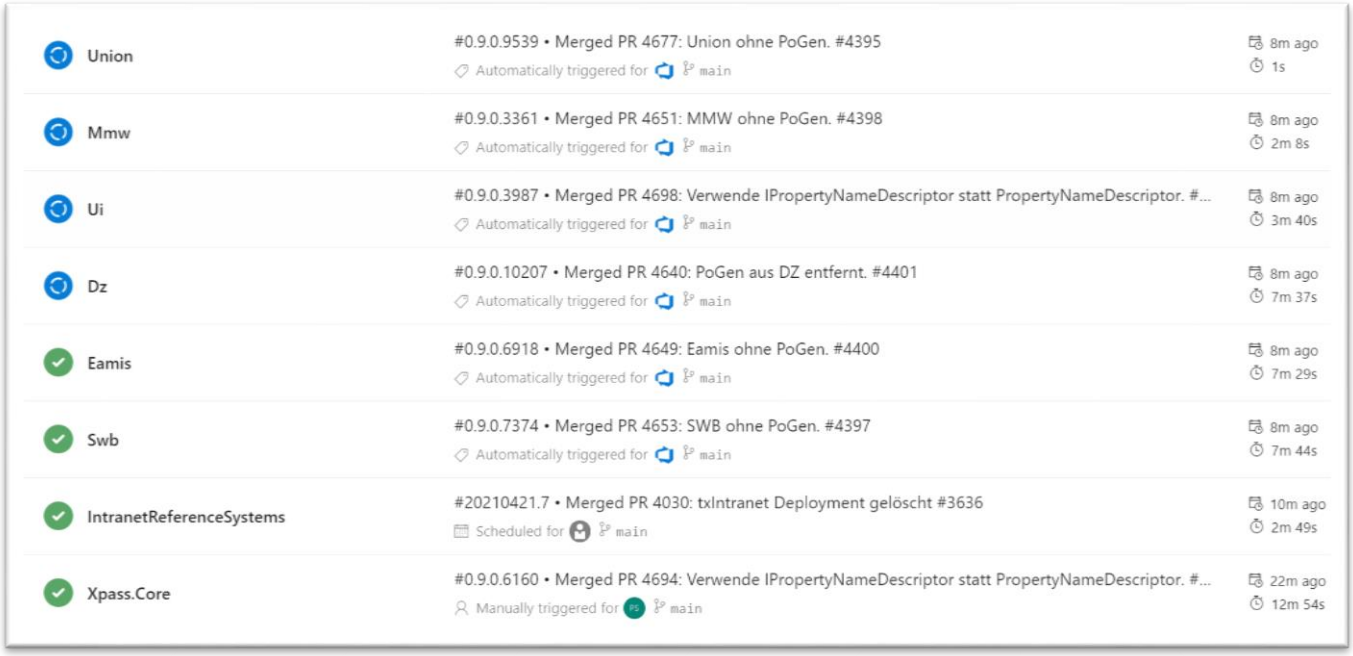



Рисунок 10 – Автоматический запуск зависимостей

Таким образом можно с легкостью управлять зависимостями различных версий сборок клиентских библиотек внутри проекта, что значительно упрощает рутинную работу менеджера проектов, т.к. до разработки сервиса, пользователю приходилось непосредственно ждать окончания сборки конвейера обработки и также знать непосредственно зависимые конвейеры обработки, которые затем нужно было запускать вручную.

# 6. ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ

## 6.1 Оценка коммерческого потенциала и перспективности проведения исследования с позиции ресурсоэффективности и ресурсосбережения

Основной задачей данного раздела является оценка перспективности разработки и планирование финансовой и коммерческой ценности конечного продукта, предлагаемого в рамках НИ.

Данный раздел, предусматривает рассмотрение следующих задач:

- Оценка коммерческого потенциала разработки;
- Планирование научно-исследовательской работы;
- Расчет бюджета научно-исследовательской работы;
- Определение ресурсной, финансовой, бюджетной эффективности исследования.

Анализируемым проектом является разработка сервис для проверки зависимостей сборок проектов на актуальность версий используемых в них внешних библиотек/модулей с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps.

Сервис должен позволяет отслеживать и с легкостью управлять зависимостями различных версий сборок клиентских библиотек внутри проекта.

Цель работы – оценка полных денежных затрат необходимых для реализации, ввода в эксплуатацию и обслуживанию универсальной веб-платформы для продажи и использования игровых и прикладных приложений.

## 6.2 Технология QuaD

Анализ конкурирующих разработок помогает вносить коррективы в научное исследование, чтобы успешнее противостоять своим конкурентам. Важно реалистично оценить сильные и слабые стороны ближайших аналогов.

Для разрабатываемого сервиса, как и для любого другого приложения в веб среде, можно выделить следующие оценочные критерии:

- Быстродействие сервиса – скорость обработки запросов;
- Возможность гибкой настройки – легкость в настройке параметров сервиса;
- Обратная связь – качество работы с клиентами;
- Безопасность – защита пользовательской информации, авторских прав;
- Надежность – обеспечение бесперебойной работы веб-приложения;
- Простота эксплуатации – удобство пользовательского интерфейса;
- Поддержка различных языков программирования – поддержка актуальных версий браузеров.

Критерии оценки веб-приложений указаны в таблице 9.

Таблица 9 – Оценочная карта для сравнения конкурентных технических решений (разработок)

| Критерии оценки | Вес критерия | Баллы | Макси-мальный балл | Относительное значение | Средневзвешенное значение |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | |
| Показатели оценки качества разработки | | | | | |
| Быстродействие сервиса | 0,2 | 100 | 100 | 1 | 20 |
| Возможность гибкой настройки | 0,3 | 80 | 100 | 0,8 | 24 |
| Обратная связь | 0,05 | 100 | 100 | 1 | 5 |
| Безопасность | 0,1 | 60 | 100 | 0,6 | 6 |
| Надежность | 0,1 | 80 | 100 | 0,8 | 8 |
| Простота эксплуатации | 0,15 | 80 | 100 | 0,8 | 12 |
| Поддержка различных языков программирования | 0,1 | 100 | 100 | 1 | 10 |
| Итого | 1 | 600 | 700 | 6 | 85 |

В соответствии с технологией QuaD каждый показатель оценивается экспертным путем по сто балльной шкале, где 1 – наиболее слабая позиция,

а 100 – наиболее сильная. Веса показателей, определяемые экспертным путем, в сумме должны составлять 1. Оценка качества и перспективности по технологии QuaD определяется по формуле:

$$П_{ср} = \sum B_i \cdot Б_i , \quad (1)$$

где $П_{ср}$ – средневзвешенное значение показателя качества и перспективности научной разработки;

$B_i$ – вес показателя (в долях единицы);

$Б_i$ – средневзвешенное значение $i$-го показателя.

Значение $П_{ср}$ позволяет говорить о перспективах разработки и качестве проведенного исследования. Если значение показателя $П_{ср}$ получилось от 100 до 80, то такая разработка считается перспективной. Если от 79 до 60 – то перспективность выше среднего. Если от 69 до 40 – то перспективность средняя. Если от 39 до 20 – то перспективность ниже среднего. Если 19 и ниже – то перспективность крайне низкая.

По результатам проведенного анализа видим, что разрабатываемая методика имеет высокие шансы на успех, следовательно, актуально создание такого сервиса.

### 6.3 SWOT-анализ

Следующим этапом является комплексный анализ внешней и внутренней среды проекта с помощью технологии SWOT, который проводится в несколько шагов.

Первый этап заключается в описании сильных и слабых сторон проекта, в выявлении возможностей и угроз для реализации проекта, которые проявились или могут появиться в его внешней среде.

Сильные стороны – это факторы, характеризующие конкурентоспособную сторону научно-исследовательского проекта.

Слабая сторона – это недостаток, упущение или ограниченность проекта, который препятствуют достижению его целей.

Возможности включают в себя любую предпочтительную ситуацию в настоящем или будущем, возникающую в условиях окружающей среды проекта, например, тенденцию, изменение или предполагаемую потребность, которая поддерживает спрос на результаты проекта и позволяет руководству проекта улучшить свою конкурентную позицию.

Угроза представляет собой любую нежелательную ситуацию, тенденцию или изменение в условиях окружающей среды проекта, которые имеют разрушительный или угрожающий характер для его конкурентоспособности в настоящем или будущем.

После того как сформулированы четыре области SWOT переходят к реализации второго этапа. В рамках данного этапа необходимо построить интерактивную матрицу проекта. Ее использование помогает разобраться с различными комбинациями взаимосвязей областей матрицы SWOT.

В рамках третьего этапа составляется итоговая матрица SWOT-анализа, которая приводится в таблице 10.

Таблица 10 – Итоговая таблица SWOT-анализа

| | **Сильные стороны:**<br>С1. Заявленная надежность и безопасность.<br>С2. Возможность гибкой настройки под требования компании.<br>С3. Бесплатен.<br>С4. Использование современных технологий в процессе разработки.<br>С5. Дешевизна разработки. | **Слабые стороны:**<br>Сл1. Наличие багов на старте.<br>Сл2. Требование технических навыков для настройки параметров сервиса;<br>Сл3. Совместимость с ограниченным количеством сервисом на старте.<br>Сл4. Отсутствие репутации на рынке.<br>Сл5. Требовательность к вычислительным ресурсам. |
|---|---|---|
| **Возможности:**<br>В1. Добавление новых функциональных возможностей.<br>В2. Повышение спроса на программный продукт. | В1С1С2С4<br>Создание новых функций, сокращающих и упрощающих работу менеджеров проектов. | В1В2Сл1Сл3Сл4<br>Проект находится на стадии тестирования<br>В2Сл2Сл3<br>Ограничение пользовательской базы в связи с навыками и требованию к технологиям<br>В2Сл5<br>предоставление серверного оборудования, удовлетворяющие потребности |

| | | |
|---|---|---|
| **Угрозы:**<br>У1. Развитие конкурентов и появление новых аналогов<br>У2. Развитая конкуренция технологий производства<br>У3. Сбои в системе или обнаружение ошибок в работе системы, требующих серьезного вмешательства в ее функционирование.<br>У4. Низкая востребованность сервиса среди пользователей. | У1С1С3С4С5<br>Отслеживание рыночной ситуации с целью внедрения актуальных и востребованных функций данного рода разработок.<br>У2С1С5<br>Невысокая цена и хорошая функциональность способствуют повышению спроса.<br>У3С1<br>Введение логов процесса работы ПО. | У1У2У3Сл3Сл4<br>Привлечение новых спонсоров и партнёров<br>У1У2Сл2Сл3Сл4<br>Исследование опыта других подобных проектов поможет избежать самого негативного сценария<br>У1У2У3У4Сл1Сл3Сл5<br>Благодаря возможности легкой модификации ПО, неожиданное несоответствие различным требованиям может быть устранено в кратчайшие сроки |

Таким образом, в результате SWOT-анализа были рассмотрены сильные и слабые стороны сервиса, очерчены возможные направления дальнейшей разработки программной системы и рассмотрены варианты минимизации влияния угроз, влияющих на работу системы. Для повышения популярности сервиса необходимо адаптировать текущую версию под более широкий набор сторонних сервисов и минимизировать количество ошибок для стабильной работы пользователя. Также повышения дальнейшей эффективности работы сервиса необходимо минимизировать угрозы, контролировать процесс создания системы на соответствие требований и постоянно проверять стабильность её работы.

### 6.4 Структура работ в рамках научного исследования

При организации процесса реализации конкретного проекта необходимо рационально планировать занятость каждого из его участников и сроки проведения отдельных работ. В данном пункте составляется полный перечень проводимых работ и определяются их исполнители.

Планирование комплекса предполагаемых работ осуществляется в следующем порядке:

– определение структуры работ в рамках научного исследования;

– определение участников каждой работы;

– установление продолжительности работ;

– построение графика проведения научных исследований.

Примерный порядок составления этапов и работ, распределение исполнителей по данным видам работ приведен в таблице 11.

Таблица 11 – Перечень этапов, работ и распределение исполнителей

| Основные этапы | № раб | Наименование работы | Исполнители работы |
|---|---|---|---|
| Разработка технического задания | 1 | Составление и утверждение темы работы | Руководитель, Инженер |
| Выбор направления исследования | 2 | Составление календарного плана-графика выполнения работы | Руководитель, Инженер |
| | 3 | Подбор и изучение литературы по теме работы | Инженер |
| | 4 | Анализ предметной области | Инженер |
| Проектирование | 5 | Проектирование инфраструктуры сервиса | Инженер |
| | 6 | Проектирование архитектуры сервиса | Инженер |
| | 7 | Проектирование базы данных | Инженер |
| Подготовка к реализации | 8 | Настройка Git, непрерывный интеграции | Инженер |
| Реализация | 9 | Реализация шаблонов развертывания инфраструктуры | Инженер |
| | 10 | Реализация структуры базы данных | Инженер |
| | 11 | Реализация сервиса | Инженер |
| Тестирование | 12 | Тестирование инфраструктуры под нагрузкой | Инженер |
| | 13 | Тестирование API сервиса | Инженер |
| | 14 | Тестирование сервиса в облачной среде | Инженер |
| Оформление отчета по работе | 15 | Согласование выполненной работы с научным руководителем | Руководитель, Инженер |
| | 16 | Выполнение других частей работы (финансовый менеджмент, социальная ответственность) | Инженер |
| | 17 | Подведение итогов, оформление работы | Инженер |

### 6.5 Определение трудоемкости выполнения работ

Трудовые затраты в большинстве случаях образуют основную часть стоимости разработки, поэтому важным моментом является определение трудоемкости работ каждого из участников научного исследования. Трудоемкость выполнения научного исследования оценивается экспертным

путем в человеко-днях и носит вероятностный характер, т.к. зависит от множества трудно учитываемых факторов. Для определения, ожидаемого (среднего) значения трудоемкости $t_{ожi}$ используется следующая формула:

$$t_{\text{ожі}} = \frac{3t_{\min i} + 2t_{\max i}}{5}, \qquad (2)$$

где $t_{ожi}$ – ожидаемая трудоемкость выполнения *i*-ой работы чел.-дн.;

$t_{\min i}$ – минимально возможная трудоемкость выполнения заданной *i* -ой работы (оптимистическая оценка: в предположении наиболее благоприятного стечения обстоятельств), чел.-дн.;

$t_{\max i}$ – максимально возможная трудоемкость выполнения заданной *i* -ой работы (пессимистическая оценка: в предположении наиболее неблагоприятного стечения обстоятельств), чел.-дн.

Исходя из ожидаемой трудоемкости работ, определяется продолжительность каждой работы в рабочих днях $T_р$, учитывающая параллельность выполнения работ несколькими исполнителями. Такое вычисление необходимо для обоснованного расчета заработной платы, так как удельный вес зарплаты в общей сметной стоимости научных исследований составляет около 65 %.

$$T_{\text{р}_i} = \frac{t_{\text{ожі}}}{\text{Ч}_i}, \qquad (3)$$

где $T_{pi}$ – продолжительность одной работы, раб. дн.;

$t_{ожi}$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн;

$\text{Ч}_i$ – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

Таблица трудоемкости выполнения работ приведена в приложении А.

### 6.6 Разработка графика проведения научного исследования

Диаграмма Ганта является наиболее удобным и наглядным способом представления графика проведения работ.

Диаграмма Ганта представляет собой отрезки, размещенные на горизонтальной шкале времени. Каждый отрезок соответствует отдельной задаче или подзадаче. Начало, конец и длина отрезка на шкале времени соответствуют началу, концу и длительности задачи.

Для построения графика Ганта, следует, длительность каждой из выполняемых работ из рабочих дней перевести в календарные дни. Для этого необходимо воспользоваться следующей формулой, для каждого исполнителя расчеты производятся индивидуально:

$$T_{\text{к}i} = T_{\text{р}i} \cdot k_{\text{кал}}, \qquad (4)$$

где $T_{\text{к}i}$ – продолжительность выполнения $i$-й работы в календарных днях;

$T_{pi}$ – продолжительность выполнения $i$-й работы в рабочих днях;

$k_{кал}$ – коэффициент календарности.

Коэффициент календарности определяется по следующей формуле:

$$k_{\text{кал}} = \frac{T_{\text{кал}}}{T_{\text{кал}} - T_{\text{вых}} - T_{\text{пр}}}, \qquad (5)$$

где $T_{\text{кал}}$ – количество календарных дней в году;

$T_{\text{вых}}$ – количество выходных дней в году;

$T_{\text{пр}}$ – количество праздничных дней в году.

Коэффициент календарности равен:

$$k_{кал} = 365 / (365 - 118) = 1.478$$

Рассчитанные значения в календарных днях по каждой работе $T_{\text{к}i}$ необходимо округлить до целого числа.

Все значения, полученные при расчетах по вышеприведенным формулам, были сведены в приложении Б.

Для описания полных поэтапных временных затрат была построена диаграмма (приложение Б).

### 6.7 Расчет материальных затрат НТИ

К данной статье расходов относится стоимость материалов, покупных изделий, полуфабрикатов и других материальных ценностей, расходуемых непосредственно в процессе выполнения работ над объектом проектирования.

Расчет материальных затрат осуществляется по следующей формуле:

$$З_{м} = (1 + k_T) \cdot \sum_{i=1}^{m} Ц_i \cdot N_{расхi} \ , \qquad (6)$$

где $m$ – количество видов материальных ресурсов, потребляемых при выполнении научного исследования;

$N_{расхi}$ – количество материальных ресурсов $i$-го вида, планируемых к использованию при выполнении научного исследования (шт., кг, м, м$^2$ и т.д.);

$Ц_i$ – цена приобретения единицы $i$-го вида потребляемых материальных ресурсов (руб./шт., руб./кг, руб./м, руб./м$^2$ и т.д.);

$k_T$ – коэффициент, учитывающий транспортно-заготовительные расходы.

Значения цен на материальные ресурсы могут быть установлены по данным, размещенным на соответствующих сайтах в Интернете предприятиями-изготовителями (либо организациями-поставщиками).

Материальные затраты, необходимые для данной разработки, заносятся в таблицу 12.

Таблица 12 – Материальные затраты

| Наименование материалов | Цена за ед., руб. | Кол-во | Сумма, руб. |
|---|---|---|---|
| Бумага для принтера, А4 | 240,00 | 1 уп. | 240,00 |
| Ручка шариковая | 20,00 | 1 шт. | 20,00 |
| Итого: | | | 260,00 |

Допустим, что коэффициент, учитывающий транспортно-заготовительные расходы составляет 15 % от отпускной цены материалов, тогда расходы на материалы с учетом коэффициента равны:

$$З_м = 1{,}15 * 260 = 299 \text{ руб.}$$

### 6.7.1 Расчет амортизации оборудования для экспериментальных работ

В данную статью включают все затраты, связанные с приобретением специального оборудования (приборов, контрольно-измерительной аппаратуры, стендов, устройств и механизмов), необходимого для проведения работ по конкретной теме. Определение стоимости спецоборудования производится по действующим прейскурантам, а в ряде случаев по договорной цене.

Для реализации проекта было использовано оборудование и программное обеспечение, затраты на которые приведены в таблице 13.

Таблица 13 – Затраты на оборудование и программное обеспечение

| Наименование оборудования | Количество единиц оборудования | Цена за 1 ед. оборудования | Затраты, руб. |
|---|---|---|---|
| Ноутбук | 1 | 80 000 | 80 000 |
| ЖК моникор | 1 | 15 000 | 30 000 |
| Клавиатура | 1 | 2 000 | 2 000 |
| Компьютерная мышь | 1 | 1 000 | 2 000 |
| Месячная подписка на портал Microsoft Azure | 1 | 11 250/мес | 101250 |
| Операционная система Windows 10 | 1 | 2700 | 2700 |
| Итого: | | | 217 950 |

Расчет амортизации проводится следующим образом:

Норма амортизации:

$$H_A = \frac{1}{n}, \qquad (7)$$

где $n$ – срок полезного использования в количестве лет.

Годовая амортизация:

$$А = И * На, \qquad (8)$$

где $И$ – итоговая сумма, тыс. руб.;

$T_{обі}$–время использования оборудования, месяцы.

Рассчитаем амортизацию для персонального компьютера, с учётом, что срок полезного использования 5 лет:

$$H_A = \frac{1}{n} = \frac{1}{5} = 0{,}2$$

Находим общую сумму амортизационных отчислений:

Годовые амортизационные отчисления:

Аг = 80 000 × 0,2 = 16 000 рублей

Ежемесячные амортизационные отчисления:

Ам = 16 000 / 12 = 1 333 рублей

Итоговая сумма амортизации основных средств для персональных компьютеров и периферии, использованных в течение 9 месяцев:

А = 1 333 × 9 = 11 997 рублей

### 6.7.2 Основная заработная плата исполнителей темы

Данная статья расходов включает заработную плату научного руководителя и инженера, в его роли выступает исполнитель проекта, а также премии, входящие в фонд заработной платы. Расчет основной заработной платы сводится в таблице 14.

Статья включает основную заработную плату работников, непосредственно занятых выполнением НТИ, и дополнительную заработную плату:

$$З_{зп} = З_{осн} + З_{доп}, \qquad (9)$$

где $З_{осн}$ – основная заработная плата;

$З_{доп}$ – дополнительная заработная плата (12-20 % от $З_{осн}$).

Основная заработная плата руководителя (лаборанта, студента) от предприятия рассчитывается по следующей формуле:

$$З_{осн} = З_{дн} \cdot T_{р}, \qquad (10)$$

где $З_{осн}$ – основная заработная плата одного работника;

$Т_р$ – продолжительность работ, выполняемых научно-техническим работником, раб. дн. (таблица 11);

$З_{дн}$ – среднедневная заработная плата работника, руб.

Среднедневная заработная плата рассчитывается по формуле:

$$З_{дн} = \frac{З_{м} \cdot М}{F_{д}}, \qquad (11)$$

где $З_м$ – месячный должностной оклад работника, руб.;

М – количество месяцев работы без отпуска в течение года:

при отпуске в 24 раб. дня М =11,2 месяца, 5-дневная неделя;

при отпуске в 56 раб. дней М=10,4 месяца, 6-дневная неделя;

$F_д$ – действительный годовой фонд рабочего времени научно-технического персонала, раб. дн.

Таблица 14 – Баланс рабочего времени

| Показатели рабочего времени | Научный руководитель | Инженер |
|---|---|---|
| Календарное число дней | 365 | 365 |
| Количество нерабочих дней | 66 | 118 |
| Потери рабочего времени на отпуск | 56 | 24 |
| Действительный годовой фонд рабочего времени | 299 | 247 |

Месячный должностной оклад работника:

$$З_{м} = З_{окл} \cdot k_{р}, \qquad (12)$$

где $З_{окл}$ – оклад, руб.;

$k_р$ – районный коэффициент, равный 1,3 (для Томска).

Научный руководитель имеет должность доцента и степень кандидата технических наук оклад составлял 35 111 руб.

Месячный оклад студента (дипломника) приравнивается к зарплате ассистента без степени и также равен 22 695 рублей.

Таблица 15 – Расчёт основной заработной платы

| Исполнители | Разряд | $k_т$ | $З_{окл}$, руб. | $k_р$ | $З_м$, руб | $З_{дн}$, руб. | $Т_р$, раб. дн. | $З_{осн}$, руб. |
|---|---|---|---|---|---|---|---|---|
| Научный руководитель | – | – | 35 111 | 1,3 | 45 644 | 1 587 | 7 | 11 109 |
| Инженер | – | – | 22 695 | | 29 503 | 1 337 | 247 | 330 239 |
| Итого $З_{осн}$ | | | | | | | | 341 348 |

### 6.7.4 Дополнительная заработная плата исполнителей темы

Затраты по дополнительной заработной плате исполнителей темы учитывают величину предусмотренных Трудовым кодексом РФ доплат за отклонение от нормальных условий труда, а также выплат, связанных с обеспечением гарантий и компенсаций (при исполнении государственных и общественных обязанностей, при совмещении работы с обучением, при предоставлении ежегодного оплачиваемого отпуска и т.д.). Расчет дополнительной заработной платы ведется по следующей формуле:

$$З_{доп} = k_{доп} \cdot З_{осн}, \qquad (13)$$

где $k_{доп}$ – коэффициент дополнительной заработной платы (на стадии проектирования принимается равным 0,12 – 0,15).

Дополнительная заработная плата представлена в таблице 16.

Таблица 16 – Расчёт дополнительной заработной платы

| Исполнитель | $k_{доп}$ | $З_{осн}$ | $З_{доп}$ |
|---|---|---|---|
| Научный руководитель | 0,12 | 11 109 | 1 333 |
| Инженер | | 330 239 | 39 629 |
| Итого | | | 40 962 |

### 6.7.5 Отчисления во внебюджетные фонды (страховые отчисления)

В данной статье расходов отражаются обязательные отчисления по установленным законодательством Российской Федерации нормам органам государственного социального страхования (ФСС), пенсионного фонда (ПФ) и медицинского страхования (ФФОМС) от затрат на оплату труда работников. Величина отчислений во внебюджетные фонды определяется исходя из следующей формулы:

$$З_{\text{внеб}} = k_{\text{внеб}} \cdot (З_{\text{осн}} + З_{\text{доп}}), \qquad (14)$$

где $k_{\text{внеб}}$ – коэффициент отчислений на уплату во внебюджетные фонды (пенсионный фонд, фонд обязательного медицинского страхования и пр.).

Для учреждений, осуществляющих образовательную и научную деятельность в 2014 году водится пониженная ставка – 27,1%.

Таблица 17 – Отчисления во внебюджетные фонды

| Исполнитель | Руководитель | Инженер |
|---|---|---|
| Основная заработная плата, руб. | 11 109 | 330 239 |
| Дополнительная заработная плата, руб. | 1 333 | 39 629 |
| Коэффициент отчислений во внебюджетные фонды | 0,271 | |
| Сумма отчислений | 3 372 | 100 234 |
| Итого | 103 606 | |

### 6.7.6 Накладные расходы

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопирование материалов исследования, оплата услуг связи, электроэнергии, почтовые и телеграфные расходы, размножение материалов и т.д. Их величина определяется по следующей формуле:

$$З_{\text{накл}} = (\text{сумма статей } 1 \div 7) \cdot k_{\text{нр}} \qquad (15)$$

где $k_{нр}$ – коэффициент, учитывающий накладные расходы. Величина коэффициента принимается равной 0,16.

$$З_{накл} = (299 + 17\,100 + 341\,348 + 40\,962 + 103\,606) * 0{,}16$$
$$= 75\,430 \text{ руб.}$$

На основании полученных данных по отдельным статьям затрат составляется калькуляция плановой себестоимости НИ по форме, приведенной в таблице 18.

Таблица 18 – Расчет бюджета затрат НТИ

| Наименование статьи | Сумма, руб. |
|---|---|
| Материальные затраты НТИ | 299 |
| Затраты на амортизацию оборудования. | 17 100 |
| Затраты по основной заработной плате исполнителей темы | 341 348 |
| Затраты по дополнительной заработной плате исполнителей темы | 40 962 |
| Отчисления во внебюджетные фонды | 103 606 |
| Накладные расходы | 80 530 |
| Бюджет затрат НТИ | 583 845 |

### 6.8 Оценка научного уровня

Важнейшим результатом реализации проекта является его научно-технический уровень, который характеризует, в какой мере выполнены работы и обеспечивается ли научно-технический прогресс в данной области.

На основе оценок новизны результатов, их ценности, масштабам реализации определяется показатель научно-технического уровня по формуле:

$$H_m = \sum_{i=1}^{n} K_i \cdot П_i , \qquad (16)$$

Где $K_i$ – весовой коэффициент *i*-го признака научно-технического эффекта;

$П_i$ – количественная оценка *i*-го признака научно-технического уровня работы.

По каждому из факторов экспертным путем при помощи нижеприведенных таблиц устанавливаются численные значения коэффициента значимости, и проставляется балльная оценка.

Таблица 19 – Признаки научно-технического эффекта

| Признак научно-технического эффекта НИР (i) | Примерное значение весового коэффициента ($K_i$) |
|---|---|
| 1.Уровень новизны | 0,6 |
| 2.Теоретический уровень | 0,4 |
| 3. Возможность реализации | 0,2 |

Таблица 20 – Количественная оценка уровня новизны НИР

| Уровень новизны разработки | Характеристика уровня новизны | Баллы |
|---|---|---|
| Принципиально новая | Результаты исследований открывают новое направление в данной области науки и техники. | 8-10 |
| Новая | По-новому или впервые объяснены известные факты, закономерности. | 5-7 |
| Относительно новая | Результаты исследований систематизируют и обобщают имеющиеся сведения, определяют пути дальнейших исследований. | 2-4 |
| Традиционная работа | Работа выполнена по традиционной методике, результаты которой носят информационный характер. | 1 |
| Не обладает новизной | Получен результат, который был ранее известен | 0 |

Таблица 21 – Количественная оценка теоретического уровня НИР

| Теоретический уровень полученных результатов | Баллы |
|---|---|
| 1.Установка закона, разработка новой теории | 10 |
| 2.Глубокая разработка проблемы, многоспектральный анализ, взаимодействия между факторами с наличием объяснений | 8 |
| 3.Разработка способа (алгоритм, программа мероприятий, устройство, вещество и т.п.) | 6 |
| 4.Элементарный анализ связей между фактами с наличием гипотезы, симплексного прогноза, классификации, объясняющей версии или практических рекомендаций частного характера. | 2 |
| 5.Описание отдельных элементарных факторов (вещей, свойств и отношений); изложение опыта, результатов измерений. | 0,5 |

Возможность реализации научных результатов представлена в таблице 22.

Таблица 22 – Возможность реализации научных результатов

| Время реализации | Баллы |
|---|---|
| В течении первых лет | 10 |
| От 5 до 10 лет | 4 |
| Более 10 лет | 2 |
| Масштабы реализации | Баллы |
| Одно или несколько предприятий | 2 |
| Отрасль (министерство) | 4 |
| Народное хозяйство | 10 |

По результатам проведения оценки НИР была составлена сводная таблица оценки научно-технического уровня, на основе которой сделан вывод об ожидаемой эффективности выполняемой НИР.

Таблица 23 – Количественная оценка признаков НИР

| Признак научно-технического эффекта НИР | Характеристика признака НИР | $K_i$ | $П_i$ |
|---|---|---|---|
| 1. Уровень новизны | Новая | 0,6 | 6 |
| 2. Теоретический уровень | Разработка способа (алгоритм, программа мероприятий, устройство, вещество и т.п.) | 0,4 | 6 |
| 3. Возможность реализации | В течении первых лет на одном предприятии | 0,2 | 12 |

Расчет НТУ и его оценка:

$$НТУ = 0,6 \cdot 6 + 0,4 \cdot 6 + 0,2 \cdot 12 = 8,4$$

Уровень научно-технического эффекта определим по таблице 24.

Таблица 24 – Оценка уровня НТЭ

| Уровень НТЭ | Коэффициент НТЭ |
|---|---|
| Низкий | 1-4 |
| Средний | 5-7 |
| Сравнительно высокий | 8-10 |
| Высокий | 11-14 |

Из таблицы видно, что разработанная система имеет сравнительно высокий уровень научно-технического эффекта.

**Заключение по разделу**

Подводя итог, можно сделать следующий вывод: разрабатываемая веб-платформа будет использоваться для дальнейшей реализации коммерческих программных продуктов, включающие в себя использование качественной графики.

Подразумевается, что в дальнейшем данный проект будет модифицироваться и расширять свой функционал, из-за чего дальнейшая себестоимость и длительность разработки увеличатся. Но дальнейшая модификация станет возможной после запуска и наращивание аудитории пользования платформой, а также после ее окупаемости первыми коммерческими проектами.

# 7. СОЦИАЛЬНАЯ ОТВЕТСВЕННОСТЬ

## Введение

Целью работы является раскрытие понятия облачных технологий, анализ продуктов на основе технологии, доступных как простым пользователям, так и бизнесу, и разработчикам. Также необходимо разработать сервис для проверки зависимостей сборок проектов на актуальность версий используемых в них внешних библиотек/модулей с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps.

Сервис должен позволяет отслеживать и с легкостью управлять зависимостями различных версий сборок клиентских библиотек внутри проекта.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить обзор доступных материалов по данной теме;
- определить основные преимущества облачных технологий;
- провести анализ ПО на основе облачных технологий;
- решить практическую задачу, состоящую в разработке сервиса на основе облачной платформы Azure DevOps.

В работе рассмотрены следующие задачи: обзор теоретической составляющей облачных технологии, её основные категории и понятия, а также преимущества и недостатки; описание предметной области, особенности уже существующих конкретных продуктов с аналогичным функционалом  и произведено их сравнение с разрабатываемым сервисом; разработка архитектуры сервиса, разработка модели базы данных, описание используемых сущностей; обоснование выбора среды разработки и языков программирования и программная реализации сервиса, создание архитектуры программного обеспечения платформы, описание ее деталей, а также написание шаблонов для развертывания инфраструктуры в среде Azure.

В проектной деятельности необходимо учитывать безопасность труда и окружающей среды. Под понятием «социальная ответственность» понимается состояние рабочего места и помещения, режим трудовой деятельности и обеспечение мероприятий по защите трудящихся в моменты чрезвычайных ситуаций. Все выше перечисленное регламентируются в соответствии с международным стандартом ICCSR26000:2011 «Социальная ответственность организации», целью которого является принятие проектных решений, исключающих несчастные случаи на производстве и негативные воздействия на окружающую среду.

## 7.1 Правовые и организационные вопросы обеспечения безопасности

### 7.1.1 Специальные (характерные для проектируемой рабочей зоны) правовые нормы трудового законодательства

Специальные ограничения работы специалиста – пользователя программой ограничиваются общими нормами трудового законодательства и СаНПиН 2.2.2/2.4.1340-03 [35].

Нормальная продолжительность рабочего времени не может превышать 40 часов в неделю.

Так как работа программиста относится к категории работ требующей постоянного взаимодействия с ПЭВМ, то рекомендуется организовывать перерывы длительностью 10 – 15 минут через каждые 40–60 минут работы. Во время таких перерывов рекомендуется выполнять комплекс упражнений для снижения нервно-эмоционального напряжения, утомления зрительного анализатора.

Законом о «правах на результаты интеллектуальной деятельности и средства индивидуализации» устанавливаются правила обеспечения безопасности авторов интеллектуальной собственности.

Данным законом регламентируются права авторов и пользователей объектов интеллектуальной собственности.

Авторские права на все виды программ для ЭВМ (в том числе на операционные системы и программные комплексы), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код, охраняются так же, как авторские права на произведения литературы.

### 7.1.2 Организационные мероприятия при компоновке рабочей зоны

Требования к организации рабочих мест пользователей, обеспечивающие безопасность при работе:

- рабочее место должно быть организовано с учетом эргономических требований согласно ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования» [19] и ГОСТ 12.2.061-81 «ССБТ. Оборудование производственное. Общие требования безопасности к рабочим местам»; [20]
- конструкция рабочей мебели (компьютерный стол, офисное кресло) должна предусматривать возможность регулировки в соответствии с индивидуальными особенностями пользователя для создания комфортных условий для выполнения работы. Вокруг ПК должно быть обеспечено свободное пространство в радиусе как минимум 60-120см.

Помещение, где выполнялась магистерская работа, имеет следующие характеристики:

- ширина рабочего помещения 5м, длина – 10 м, высота – 3 м
- площадь – 50 м2
- объём помещения – 150 м3
- имеется естественная вентиляция: двери, окна
- искусственное освещение
- естественное освещение

В данном помещении оборудовано четыре рабочих места, одновременно в работе обычно задействованы 2 человека. Следовательно, в среднем на одного сотрудника приходится не менее 75 $м^3$ объема

помещения и не менее 25 $м^2$ площади это удовлетворяет требованиям санитарных норм. По санитарным нормам для одного работника должны быть предусмотрены площадь величиной не менее 6 $м^2$ и объем не менее 24 $м^3$, с учетом максимального числа одновременно работающих в смену. Государственными стандартами и правовыми нормами обеспечения безопасности предусмотрена рациональная организация труда в течение смены, которая предусматривает:

- продолжительность рабочей смены, не превышающей 8 часов;
- длительность обеденного перерыва не меньше 30 минут;
- установление двух регламентируемых перерывов (не меньше 20 минут после 1-2 часов работы и не меньше 40 минут после 2 часов работы).

Обязательно должен быть предусмотрен предварительный медосмотр, который осуществляется при приеме на работу, и периодические медосмотры. Также перед приемом на работу каждый сотрудник должен пройти инструктаж по технике безопасности, а в дальнейшем с работником должен быть проведен инструктаж по электробезопасности и охране труда.

### 7.2 Производственная безопасность

При разработке платформы отрицательным фактором является большой объем работы с ПК, поэтому важным критерием безопасности является организация рабочего места и режима трудовой деятельности.

Офисные сотрудники подвергаются в основном физическим и психофизиологическим факторам. В таблице 1 представлены все вредные и опасные факторы и их классификация в соответствие с нормативными документами.

Перечень опасных и вредных факторов, влияющих на сотрудников в заданных условиях деятельности, представлен в Таблице 25.

Таблица 25. Классификация вредных и опасных факторов

| Факторы (ГОСТ 12.0.003-2015) | Этапы работ | | | Нормативные документы |
|---|---|---|---|---|
| | Разработано | Изготовлено | Эксплуатировано | |
| Отклонение показателей микроклимата | + | + | + | СанПиН 2.2.4.548–96. Гигиенические требования к микроклимату производственных помещений. [21]<br>ГОСТ 12.1.005-88 ССБТ. Общие санитарно-гигиенические требования к воздуху рабочей зоны. [22] |
| Недостаточная освещенность рабочей зоны | + | + | + | СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 2305-95 [23]<br>СанПиН 2.2.1/2.1.1.1278–03. Гигиенические требования к естественному, искусственному и совмещённому освещению жилых и общественных зданий. [24] |
| Превышение уровня шума | | + | | ГОСТ 12.1.0032014 ССБТ. Шум. Общие требования безопасности. [25] |

Продолжение таблицы 25. Классификация вредных и опасных факторов

| Факторы (ГОСТ 12.0.003-2015) | Этапы работ | | | Нормативные документы |
|---|---|---|---|---|
| | Разработка | Изготовление | Эксплуатация | |
| Опасные и вредные производственные факторы, связанные с электромагнитными полями. | + | + | + | СП 2.4.3648-20 «Санитарно-эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодежи». [26]<br>СанПиН 2.2.4.3359-16 "Санитарноэпидемиологические требования к физическим факторам на рабочих местах" [27]<br>ГОСТ 12.1.006-84 ССБТ. Электромагнитные поля радиочастот. Общие требования безопасности. [28] |
| Психофизиологические факторы | + | + | + | ГОСТ 12.0.002-2014 Система стандартов безопасности труда (ССБТ). Термины и определения. [29] |
| Повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека | + | + | + | ГОСТ 12.4.01189 ССБТ «Средства защиты работающих. Классификация». [30]<br>ГОСТ 12.0.00374 ССБТ «Опасные и вредные производственные факторы. классификация» [31] |

### 7.2.1 Отклонение показателей микроклимата

В ГОСТ 12.1.005-88 представлены гигиенические нормативы на параметры микроклимата в рабочем помещение. Микроклимат определяется сочетаниями влажности, температуры воздуха, окружающих поверхностей и скорости движения воздуха действующих на организм человека В комплексные планы мероприятий по охране труда обязательно включаются мероприятия по оптимизации микроклиматических показателей до нормативных значений. Согласно СанПиН 2.2.4.548-96 выполняемая работа относится к категории легкая (1б) – интенсивность энергозатрат в пределах 121-150 ккал/час (140-174 Вт), это работы сидя, стоя или связанные с ходьбой с некоторым физическим напряжением

Таблица 26. Оптимальные величины показателей микроклимата на рабочих местах производственных помещений (СанПиН 2.2.4.548-96)

| Период года | Относительная влажность воздуха, % | Скорость движения воздуха, м/с | Температура воздуха, $^0$C | Температура поверхностей, $^0$C |
|---|---|---|---|---|
| Холодный | 60-40 | 0,1 | 21 - 23 | 20 - 24 |
| Теплый | 60-40 | 0,1 | 23 - 25 | 22 - 26 |

Таблица 27. Допустимые величины показателей микроклимата

| Период года | Относительная влажность воздуха, % | Скорость движения воздуха, м/с | | Температура воздуха | | Температура поверхностей, ºС |
|---|---|---|---|---|---|---|
| | | для диапазона температур воздуха ниже оптимальных величин не более | для диапазона температур воздуха выше оптимальных величин не более | диапазон ниже оптимальных величин | диапазон выше оптимальных величин | |
| Холодный | 15 - 75 | 0,1 | 0,2 | 19,0 - 20,9 | 23,1 - 24,0 | 18,0 - 25,0 |
| Теплый | 15 - 75 | 0,1 | 0,2 | 20,0 - 21,9 | 24,1 - 28,0 | 19,0 - 29,0 |

В помещении, где велась разработка, температура поверхностей и температура воздуха составляет 20$^0$C и 23$^0$C соответственно, а влажность

воздуха 40%; а в теплый период температура поверхностей и температура воздуха - $24^0$С и $26^0$С соответственно. Сравнивая со значениями из таблицы, отклонений от норм не выявлено.

### 7.2.2 Недостаточная освещенность рабочей зоны

Освещение оказывает влияние на общее самочувствие и настроение, определяет эффективность труда. Нерационально организованное освещение может явиться причиной травматизма: недостаточно освещенные опасные зоны, слепящие источники света и блики от них, резкие тени и пульсации освещенности ухудшают видимость и могут вызвать неадекватное восприятие наблюдаемого объекта.

В рабочем помещении должно быть, как естественное, так и искусственное освещение. Естественное освещение обеспечивается за счет оконных проемов, коэффициент искусственного освещения (КЕО) которых должен быть не менее 1,2% в местах, где имеется снежный покров и не менее 1,5% на остальной территории. Естественное освещение, свет из окна, должно падать с левой стороны от сотрудника. В помещении установлено два окна размером 3 на 1,5 метра в наружных стенах с регулируемыми жалюзи. Нормируемые показатели естественного, искусственного и совмещенного освещения в соответствии с СанПиН 2.2.1/2.1.1.1278-03 указаны в таблице 28.

Таблица 28. Нормируемые показатели естественного, искусственного и совмещенного освещения

| Помещения | Рабочая поверхность и плоскость нормирования КЕО и освещенности и высота плоскости над полом, м | Естественное освещение | | Совмещенное освещение | Искусственное освещение | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | КЕО e_н, % | | КЕО e_н, % | Освещенность, лк | | | | | |
| | | При верхнем или комбинированном освещении | При боковом освещении | При верхнем или комбинированном освещении | При боковом освещении | При комбинированном | | При общем освещении | Показатель дискомфорта, М, не более | Коэффициент пульсации освещенности , К_п,%,не более |
| | | | | | | всего | от общего | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Кабинеты, рабочие комнаты | Г – 0,8 | 3,0 | 1,0 | 1,8 | 0,6 | 400 | 200 | 300 | 40 | 15 |
| Помещения для работы с дисплеями и видеотерминалами, залы ЭВМ | Г–0,8 Экран монитора: В – 1,2 | 3,5 | 1,2 | 2,1 | 0,7 | 500 | 300 | 400 | 15 | 10 |

В офисах для организации искусственного освещения рекомендуется применять светильники типа ЛПО36, ЛПО5, ЛПО13, ЛСО4, ЛПО34, ЛПО31 с люминесцентными лампами типа ЛБ. Также допускается применение светильников местного освещения с лампами накаливания. Светильники должны располагаться прямыми или прерывающимися линиями так, чтобы они были параллельны линии зрения сотрудника за компьютером. Защитный угол светильников должен быть не менее 40 градусов. В случае, когда естественного освещения недостаточно, используется общее искусственное освещение. Основными источниками искусственного освещения используются лампы белого и дневного света ЛБ-20 и ЛД-20.

Произведем расчет освещения производственного помещения. Рассматриваемое помещение имеет светлый цвет потолков и стен, серое покрытие пола. Длина помещения (a) – 10 м., ширина (b) – 5 м., высота (h) –

3 м. В качестве источника света используются светильники, каждый из которых содержит по 4 люминесцентных ламп мощностью 18 Вт.; общая яркость светового потока (Ф) 1200 Лм. Помещение предназначено для работы за персональным компьютером, поэтому нормой освещенности (Е) для него согласно СНиП 23-05-95 [32] станет 300 Лк, рабочая плоскость стола находится на расстоянии (h1) 0,8 м. над уровнем пола, коэффициент запаса (Кз) равняется 1,4, а приблизительные коэффициенты отражения, согласно таблице 1.9.3 из «Пособие к МГСН 2.06-99 Расчет и проектирование искусственного освещения помещений общественных зданий»: для побеленного потолка – 0,7; для побеленных стен при незанавешенных окнах – 0,5. Сначала находим площадь помещения (S): 5*10=50 м$^2$. Далее находим индекс помещения по формуле 15:

$$\varphi = \frac{S}{(h-h_1)*(a+b)} = \frac{50}{(3-0.8)*(10+5)} = 1{,}5 \qquad (17)$$

Теперь на основании показателей отражения поверхностей и высчитанного индекса можно из таблицы определить коэффициент использования (U). В данном случае он равняется 57%. Определим необходимое количество светильников по формуле 16:

$$N = \frac{E*S*K_з}{U*n*\Phi_л} = \frac{300*50*1{,}4}{0{,}57*4*1200} = 7{,}67 \approx 8 \qquad (18)$$

В помещении, в котором проводилась работа, используются рядно расположенные потолочные светильники с люминесцентными лампами.

Проведенные расчеты показали, что минимальное число светильников должно быть равно 8. В результате анализа освещенности рабочего места отклонений от норм выявлено не было, так как в помещении находится 12 светильников. Уровень освещенности соответствует нормам в разные периоды светового дня.

### 7.2.3 Превышение уровня шума

При постоянном нахождение в помещение где уровень шума более 85 децибел, могут наблюдаться нарушения слуха. Для помещения, в котором велась разработка, основными источниками шума являются

расположенные в помещении компьютеры и кондиционер. Уровни шума для различных категорий рабочих мест служебных помещений регламентирует ГОСТ 12.1.003-2014. «ССБТ. Шум. Общие требования безопасности».

Помещения, в которых для работы используют компьютеры не должны соседствовать с помещениями, в которых уровни шума превышают нормируемые значения. В помещениях, которые оборудованы компьютерами, которые являются основным источником шума, уровень шума на рабочем месте должен быть не более 50 дБ.

Рассматриваемое помещение по уровню производственных шумов, не выходит за рамки допустимых значений. Уровень шума менее 50 дБ.

### 7.2.4 Опасные и вредные производственные факторы, связанные с электромагнитными полями

Работая за компьютером, сотрудник подвергается воздействию электромагнитного и электростатического полей. Создаваемое персональным компьютером электромагнитное излучение имеет электрическую (Е) и магнитную (Н) составляющие, а также сложный спектральный состав с диапазоном частот от 0 до 1000 МГц. Основным источником электромагнитных излучений является ЖК монитор, имеющие низкий уровень электромагнитного излучения.

СанПиН 2.2.4.3359-16 определяет нормы допустимых уровней напряженности электрических полей. Они зависят от времени пребывания человека в контролируемой зоне. Время допустимого пребывания в рабочей зоне в часах рассчитывается по формуле Т=50/Е–2. Если напряженность электрического поля лежит в диапазоне 20–25 кВ/м, то работа не может продолжаться более 10 минут. При напряженности, не превышающей 5 кВ/м деятельность людей в рабочей зоне может осуществляться в течение 8 часов.

СанПиН 2.2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»,

регламентирующий безопасные уровни излучений. В таблицах 29-30 представлены предельно-допустимые уровни напряженности на рабочих местах и допустимые уровни электромагнитных полей.

Таблица 29. Предельно-допустимые уровни напряженности на рабочих местах [33]

| Время воздействия за рабочий день, мин | Условия воздействия | | | |
|---|---|---|---|---|
| | Общее | | локальное | |
| | ПДУ | ПДУ магнитной | ПДУ | ПДУ магнитной |
| 0-10 | 24 | 30 | 40 | 50 |
| 11-60 | 16 | 20 | 24 | 30 |
| 61-480 | 8 | 10 | 12 | 15 |

Таблица 30. Допустимые уровни электромагнитных полей [33]

| Наименование параметра | |
|---|---|
| Напряженность электромагнитного поля на расстоянии 50 см вокруг дисплея до электрической составляющей, В/м, не более:<br><br>в диапазоне частот 5 Гц –2 кГц<br><br>в диапазоне частот 2 – 400 кГц | <br><br>25<br><br>2,5 |
| Плотность магнитного потока на расстоянии 50 см вокруг дисплея, нТл, не более:<br>в диапазоне частот 5 Гц - 2 кГц<br><br>в диапазоне частот 2– 400 кГц | <br><br>250<br><br>25 |
| Поверхностный электростатический потенциал, В, не более | 500 |

Для снижения уровня излучений проводятся следующие мероприятия:

- применение средств индивидуальной защиты, направленных на экранирование пользователя ПК целиком или отдельных частей его тела;
- употребление профилактических напитков;
- использование других технических средств защиты от электромагнитных излучений;

- сертификация ПК и аттестация рабочих мест;
- применение фильтров и экранов;
- организационно-технические мероприятия.

В рассматриваемом помещении, уровень напряженности электромагнитного поля не превышает предельно-допустимые значения, индивидуальная защита пользователей не требуется.

### 7.2.5 Психофизиологические факторы

Во время длительной работы за компьютером сотрудник также подвергается воздействию психофизиологических факторов, таких как эмоциональные перегрузки, умственное перенапряжение, монотонность труда и другие, определенны в ГОСТ 12.0.002-2014 и ГОСТ 12.0.003-2015 [34].

Эмоциональные перегрузки способны вызвать изменения функционального состояния центральной нервной системы, что негативно отражается на состоянии организма в целом. Они могут быть вызваны необходимостью выполнения большого объема работы, конфликтными или стрессовыми ситуациями. Умственное перенапряжение может наступать вследствие отсутствия необходимого времени на отдых после продолжительной работы, нарушения режима сна или режима питания. Оно может накапливаться и приводить к возникновению заболеваний.

Отличительными признаками монотонной работы служат однообразие рабочих действий, их многократное повторение и небольшая длительность. Таковой является работа за компьютером. В результате работающий теряет интерес к работе, и у него возникает состояние «производственной скуки». Монотонная работа отрицательно сказывается на эффективности производства: ухудшаются экономические показатели, повышается аварийность, травматизм, растет текучесть кадров. Для снижения эмоциональных перегрузок и умственных перенапряжений предусмотрены перерывы в работе, возможность выбора удобного времени для выполнения работы.

### 7.2.6 Повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека

Помещение, в котором расположены персональные компьютеры, относится к помещениям без повышенной опасности, так как согласно ГОСТ 12.0.00374 отсутствуют следующие факторы:

- высокая температура;
- токопроводящая пыль;
- токопроводящие полы;
- сырость;
- возможность одновременного прикосновения человека к имеющим соединение с землёй металлоконструкциям зданий, технологическим аппаратам и механизмам, металлическим корпусам электрооборудования.

Мероприятия, направленные на предотвращение возможности поражения электрическим током, включают в себя следующее:

- при выполнении монтажных работ необходимо использовать только исправно работающий инструмент, аттестованный службой КИПиА;
- заземление корпусов приборов и инструментов, которое поможет защитить от поражения электрическим током, который может возникнуть между корпусом приборов и инструментом при пробое сетевого напряжения на корпус;
- запрет на выполнение работ на задней панели при включенном сетевом напряжении;
- выполнение работ по устранению неисправностей должно производиться компетентными людьми;
- нужно постоянно наблюдать за исправностью электропроводки и в случае обнаружения неисправностей незамедлительно принимать действия по их устранению.

Согласно ГОСТу 12.4.01189 «Средства защиты работающих. Общие требования и классификация» к средствам защиты от повышенного уровня статического электричества относятся:

- заземляющие устройства;
- нейтрализаторы;
- увлажняющие устройства;
- антиэлектростатические вещества;
- экранирующие устройства.

### 7.3 Экологическая безопасность

Охрана окружающей среды заключается в устранении отходов жизнедеятельности человека и бытового мусора.

Если персональные компьютеры теряют свою работоспособность, их списывают и отправляют на специализированный склад, на котором уже принимаются меры по утилизации техники и комплектующих.

По статистике вышедшие из строя люминесцентные лампы являются одним из самых распространенных источников ртутного загрязнения. Помимо стекла и алюминия каждая лампа содержит приблизительно 60 мг ртути, поэтому отработавшие люминесцентные лампы являются опасным источником токсичных веществ.

Утилизация таких ламп заключается в их передаче перерабатывающим предприятиям, которые имеют специальное оборудование для переработки вредных ламп в безвредное сырье – сорбент, которое может являться материалом для других производств. Согласно ГОСТ Р 57740-2017 [35] и ГОСТ Р 51768-2001 [36] отработанные люминесцентные лампы относятся к отходам, которые собираются и сортируются отдельно, поэтому их утилизация и хранение должны отвечать определенным требованиям.

### 7.4 Безопасность в чрезвычайных ситуациях

Для помещения, в котором ведется разработка, наиболее вероятно возникновение такой ЧС как пожар, который может возникнуть при

замыкании электропроводки оборудования, обрыве проводов или же при несоблюдении мер пожарной безопасности. К противопожарным мероприятиям в помещении относятся следующие:

1) помещение должно быть оборудовано средствами тушения пожара, такими как огнетушители, стенд с противопожарным инвентарем, ящик с песком; средствами связи; электрическая проводка осветительных приборов и электрооборудования должна быть в исправном состоянии

2) каждый сотрудник должен знать месторасположение средств тушения пожара и средств связи; знать номера телефонов экстренных служб для оповещения о пожаре; уметь использовать средства пожаротушения

Рассматриваемое помещение оснащено средствами пожаротушения в соответствии с нормами:

1) огнетушитель пенный ОП-10 – 1 шт
2) огнетушитель углекислотный ОУ-5 – 1 шт

В помещении и на этаже присутствуют следующие средства оповещения:

- световая индикация направления движения к выходу в коридорах этажа;
- звуковая индикация, которая представляет собой систему оповещения о пожаре через громкоговоритель;
- пассивные датчики задымленности.

Чтобы минимизировать вероятность возникновения пожара нужно своевременно проводить профилактические работы, направленные на устранение возможных источников возникновения пожара, такие как:

- систематическое наблюдение за состоянием электропроводки;
- выключение питания оборудования при завершении работы и покидании рабочего места;
- периодическое проведение инструктажа по пожаробезопасности для персонала.

При возникновении пожара должна сработать система пожаротушения, издав предупредительные сигналы, и передав на пункт пожарной станции сигнал о ЧС, в случае если система не сработала, по каким-либо причинам, необходимо нажать тревожную кнопку или самостоятельно произвести вызов пожарной службы по телефону 101, сообщить место возникновения ЧС и ожидать приезда специалистов.

**Вывод по разделу**

В данном разделе был произведен анализ рабочего помещения, анализ вредных и опасных факторов и методы минимизации их воздействия на человеческое здоровье. Были рассмотрены нормативы микроклимата, освещения, шума, электробезопасности помещения. Исходя из полученных данных можно утверждать, что оно соответствует требованиям законодательства РФ. Также были рассмотрены аспекты экологической, производственной безопасности, безопасности в чрезвычайных ситуациях (на примере пожароопасности).

## Заключение

Целью данной работы являлось раскрытие понятия облачных технологий, анализ продуктов на основе технологии и непосредственно производилась разработка сервиса для проверки зависимостей сборок проектов на актуальность версий используемых в них внешних библиотек/модулей с применением конвейеров непрерывной интеграции в экосистеме Azure DevOps.

В ходе данной работы был проведен анализ предметной области, сравнение с уже существующими на рынке аналогами.

Проведен этап проектирования для ясного осознания внутренней структуры сервиса. Проработана спецификация требований к сервису, что позволило приступить к этапу реализации.

В программной среде Visual Studio 2019 разработаны шаблоны для развёртывания инфраструктуры сервиса с применением облачных технологий посредствам кода и реализован необходимый сервис.

Данная разработка позволила компании значительно сократить время- и трудозатраты на развёртывание приложений в облачной среде Microsoft Azure.

Также в ходе работы рассмотрены вопросы, касающиеся экономической значимости и эффективности разработанной системы, сформирован бюджет научных исследований, спланированы этапы работ, проведена оценка сравнительной эффективности внедрения.

Проанализированы опасные и вредные факторы труда, оказывающие влияние на производственную деятельность программиста, рассмотрены вопросы техники пожарной безопасности, правила поведения во время чрезвычайных ситуаций и способы их предотвращения. Рассмотрены вопросы правильной организации рабочего места программиста, а также режим труда и отдыха на рабочем месте.

Также был произведен перевод одного из разделов дипломной работы на иностранный язык.

## Conclusion

The purpose of this work was to reveal the concept of cloud technologies, analyze products based on the technology, and directly develop a service for checking the dependencies of project builds for the relevance of the versions of external libraries/modules used in them, using continuous integration pipelines in the Azure DevOps ecosystem.

This work include an analysis of the subject area was carried out and comparison with existing analogues on the market.

The design stage is carried out for a clear understanding of the internal structure of the service. The specification of the requirements for the service was worked out, which allowed us to start the implementation stage.

In the Visual Studio 2019 software environment, templates have been developed for deploying the service infrastructure using cloud technologies through code and the service is implemented.

This development has allowed the company to significantly reduce the time and labor costs for deploying applications in the Microsoft Azure cloud environment.

Also in the course of the work, issues related to the economic significance and effectiveness of the developed system were considered, the budget for scientific research was formed, the stages of work were planned, and the comparative effectiveness of implementation was evaluated.

The article analyzes the dangerous and harmful labor factors that affect the production activity of the programmer, considers the issues of fire safety, rules of conduct during emergencies and ways to prevent them. The questions of the correct organization of the programmer's workplace, as well as the mode of work and rest in the workplace are considered.

**Список используемых источников**


1. Advantages of Cloud Computing and How Your Business Can Benefit From Them / [Электронный ресурс] // URL: https://www.skyhighnetworks.com/cloud-securityblog/11-advantages-of-cloud-computing-and-how-your-business-canbenefit-from-them/ (дата обращения: 04.02.2021)
2. Введение в Cloud Computing / [Электронный ресурс] // URL: https://intuit.ru/studies/courses/677/533/lecture/12019?page=1 (дата обращения: 04.02.2021).
3. Общие сведения об облачных вычислениях / [Электронный ресурс] // https://intuit.ru/studies/courses/12160/1166/lecture/19342 (дата обращения: 04.02.2021).
4. IaaS, что это такое? PaaS, SaaS, для чего они нужны? / [Электронный ресурс] // https://1cloud.ru/services/private-cloud/iaas-paas-saas (дата обращения: 04.02.2021).
5. Что такое XaaS? Различие IaaS, SaaS, PaaS. / [Электронный ресурс] // https://www.ispsystem.ru/news/xaas#:~:text=cloud_dif (дата обращения: 04.02.2021).
6. Платформа Windows Azure / [Электронный ресурс]. – http://www.windowsazure.com/ru-ru/ (дата обращения: 04.02.2021).
7. Обзор сервисов Azure / [Электронный ресурс] – Режим доступа: https://azure.microsoft.com/ru-ru/services/ (дата обращения: 04.02.2021).
8. Azure DevOps / [Электронный ресурс] – Режим доступа: https://azure.microsoft.com/ru-ru/overview/what-is-devops/ (дата обращения: 04.02.2021).
9. DevOps методология / [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/DevOps (дата обращения: 04.02.2021).
10. Что такое DevOps Pipeline?/ [Электронный ресурс] – Режим доступа: https://dinarys.com/ru/blog/devops-pipeline (дата обращения: 20.02.2021).

11. Выполнение конвейера и триггеры в фабрике данных Azure / [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/azure/data-factory/concepts-pipeline-execution-triggers (дата обращения: 20.02.2021).

12. Automated dependency updates / [Электронный ресурс] – Режим доступа: https://dependabot.com/ (дата обращения: 20.02.2021).

13. WhiteSource Renovate / [Электронный ресурс] – Режим доступа: https://www.whitesourcesoftware.com/free-developer-tools/renovate?ref=producthunt (дата обращения: 20.02.2021).

14. Бессерверные приложения: Архитектура, шаблоны и реализация в Azure / [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/architecture/serverless/ (дата обращения: 20.02.2021).

15. Топ 10 облачных платформ для бизнеса / [Электронный ресурс] – Режим доступа: http://www.livebusiness.ru/news/8937/. (дата обращения: 20.02.2021).

16. MS SQL Server / [Электронный ресурс] – URL: https://www.microsoft.com/en-us/sql-server/sql-server-2019 (дата обращения: 20.02.2021).

17. Что такое шаблоны ARM? / [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/azure/azure-resource-manager/templates/overview (дата обращения: 20.02.2021).

18. СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы: Санитарноэпидемиологические правила и нормы». – М.: Федеральный центр госсанэпиднадзора Минздрава России, 2003. – 54 с

19. ГОСТ 12.2.032-78 ССБТ «Рабочее место при выполнении работ сидя. Общие эргономические требования». [Электронный ресурс]. URL: https://gosthelp.ru/text/GOST12203278SSBTRabocheem.html. (Дата обращения: 24.04.2021).

20. ГОСТ 12.2.061-81 «ССБТ. Оборудование производственное. Общие требования безопасности к рабочим местам». [Электронный ресурс]. URL: https://gosthelp.ru/text/GOST12206181SSBTOborudova.html. (Дата обращения: 24.04.2021).

21. СанПиН 2.2.4.548-96 «Гигиенические требования к микроклимату производственных помещений. Санитарные правила и нормы». [Электронный ресурс]. URL: https://gosthelp.ru/text/SanPiN22454896Gigieniches.html. (Дата обращения: 24.04.2021).

22. ГОСТ 12.1.005-88 «Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны». [Электронный ресурс]. URL: https://gosthelp.ru/gost/gost1583.html. (Дата обращения: 24.04.2021).

23. СП 52.13330.2016 «Естественное и искусственное освещение. Актуализированная редакция СНиП 2305-95». [Электронный ресурс]. URL: https://docs.cntd.ru/document/456054197. (Дата обращения: 24.04.2021).

24. СанПиН 2.2.1/2.1.1.1278-03 «Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий» год». [Электронный ресурс]. URL: https://gosthelp.ru/text/SanPiN221211127803Gigieni.html. (Дата обращения: 24.04.2021).

25. ГОСТ 12.1.0032014 «ССБТ. Шум. Общие требования безопасности». [Электронный ресурс]. URL: https://docs.cntd.ru/document/1200118606. (Дата обращения: 24.04.2021).

26. СП 2.4.3648-20 «Санитарно-эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодежи». [Электронный ресурс]. URL: https://docs.cntd.ru/document/566085656. (Дата обращения: 24.04.2021).

27. СанПиН 2.2.4.3359-16 «Санитарноэпидемиологические требования к физическим факторам на рабочих местах». [Электронный

ресурс]. URL: https://docs.cntd.ru/document/420362948. (Дата обращения: 24.04.2021).

28. ГОСТ 12.1.006-84 ССБТ «Электромагнитные поля радиочастот. Общие требования безопасности». [Электронный ресурс]. URL: https://docs.cntd.ru/document/5200272. (Дата обращения: 24.04.2021).

29. «Система стандартов безопасности труда». [Электронный ресурс]. URL: https://docs.cntd.ru/document/1200125989. (Дата обращения: 24.04.2021).

30. ГОСТ 12.4.01189 ССБТ «Средства защиты работающих. Классификация». [Электронный ресурс]. URL: https://docs.cntd.ru/document/1200000277. (Дата обращения: 24.04.2021).

31. ГОСТ 12.0.00374 ССБТ «Опасные и вредные производственные факторы. классификация». [Электронный ресурс]. URL: https://docs.cntd.ru/document/5200224. (Дата обращения: 24.04.2021).

32. СНиП 23-05-95 «Естественное и искусственное освещение». [Электронный ресурс]. URL: https://gosthelp.ru/text/SNiP230595Estestvennoeiis.html. (Дата обращения: 24.04.2021).

33. ГОСТ 12.1.002-84 ССБТ «Электрические поля промышленной частоты. Допустимые уровни напряженности и требования к проведению контроля на рабочих местах». [Электронный ресурс]. URL: https://gosthelp.ru/text/GOST12100284SSBTElektrich.html. (Дата обращения: 24.04.2021).

34. ГОСТ 12.0.003-2015 «Система стандартов безопасности труда. Опасные и вредные производственные факторы». [Электронный ресурс]. URL: https://docs.cntd.ru/document/1200136071. (Дата обращения: 24.04.2021).

35. ГОСТ Р 57740-2017 «Требования к приему, сортировке и упаковыванию опасных твердых коммунальных отходов». [Электронный

ресурс]. URL: https://docs.cntd.ru/document/1200156943. (Дата обращения: 24.04.2021).

36. ГОСТ Р 51768-2001 «Методика определения ртути в ртутьсодержащих отходах». [Электронный ресурс]. URL: https://docs.cntd.ru/document/1200025450. (Дата обращения: 24.04.2021).

## Приложение А – Таблица трудоемкости выполнения работ

| Название работы | Трудоёмкость работ | | | | | | Длительность работ в рабочих днях $T_{рi}$ | | Длительность работ в календарных днях $T_{кi}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $t_{min}$, чел-дни | | $t_{max}$, чел-дни | | $t_{ожi}$, чел-дни | | | | | |
| | И | НР | И | НР | И | НР | И | НР | И | НР |
| Составление и утверждение темы работы | 1 | 0.5 | 3 | 1.5 | 1.8 | 0.9 | 0.9 | 0.45 | 1.18 | 0.67 |
| Составление календарного плана-графика выполнения работы | 5 | - | 7 | - | 5.8 | - | 5.8 | - | 8.57 | - |
| Подбор и изучение литературы по теме работы | 7 | - | 14 | - | 9.8 | - | 9.8 | - | 14.48 | - |
| Анализ предметной области | 14 | - | 21 | - | 16.8 | - | 16.8 | - | 24.83 | - |
| Проектирование инфраструктуры сервиса | 15 | - | 21 | - | 17.4 | - | 17.4 | - | 25.71 | - |
| Проектирование архитектуры сервиса | 20 | - | 31 | - | 24.4 | - | 24.4 | - | 36.06 | - |
| Проектирование базы данных | 5 | - | 7 | - | 5.8 | - | 5.8 | - | 8.57 | - |

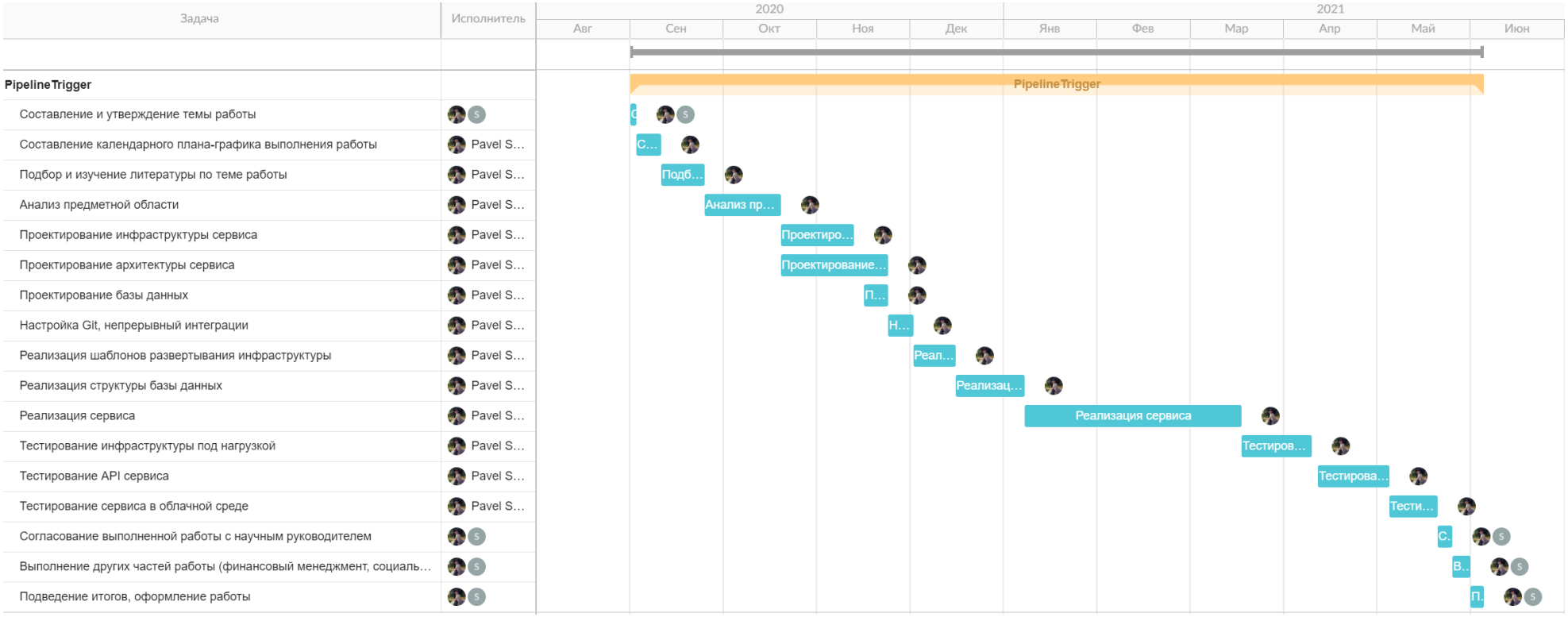
## Продолжение приложения А – Таблица трудоемкости выполнения работ

| Название работы | Трудоёмкость работ | | | | | | Длительность работ в рабочих днях $T_{рi}$ | | Длительность работ в календарных днях $T_{кi}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $t_{min}$, чел-дни | | $t_{max}$, чел-дни | | $t_{ожi}$, чел-дни | | | | | |
| | И | НР | И | НР | И | НР | И | НР | И | НР |
| Настройка Git, непрерывный интеграции | 5 | - | 7 | - | 5.8 | - | 5.8 | - | 8.57 | - |
| Реализация шаблонов развертывания инфраструктуры | 7 | - | 14 | - | 9.8 | - | 9.8 | - | 14.48 | - |
| Реализация структуры базы данных | 14 | - | 21 | - | 16.8 | - | 16.8 | - | 24.83 | - |
| Реализация сервиса | 42 | - | 63 | - | 50.4 | - | 50.4 | - | 74.49 | - |
| Тестирование инфраструктуры под нагрузкой | 14 | - | 21 | - | 16.8 | - | 16.8 | - | 24.83 | - |
| Тестирование API сервиса | 14 | - | 21 | - | 16.8 | - | 16.8 | - | 24.83 | - |

## Продолжение приложения А – Таблица трудоемкости выполнения работ

| Название работы | Трудоёмкость работ | | | | | | Длительность работ в рабочих днях $T_{pi}$ | | Длительность работ в календарных днях $T_{ki}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $t_{min}$, чел-дни | | $t_{max}$, чел-дни | | $t_{ожi}$, чел-дни | | | | $t_{min}$, чел-дни | |
| | И | НР | И | НР | И | НР | И | НР | И | НР |
| Тестирование сервиса в облачной среде | 10 | - | 14 | - | 11.6 | - | 11.6 | - | 17.14 | - |
| Согласование выполненной работы с научным руководителем | 2 | 2 | 3 | 3 | 2.4 | 2.4 | 1.2 | 1.2 | 1.77 | 1.77 |
| Выполнение других частей работы (финансовый менеджмент, социальная ответственность) | 2 | - | 5 | - | 3.2 | - | 3.2 | - | 4.73 | - |
| Подведение итогов, оформление работы | 2 | 2 | 5 | 5 | 3.2 | 3.2 | 1.6 | 1.6 | 2.36 | 2.36 |

## Приложение Б – Диаграмма Ганта для определения графика работ подэтапов за период дипломирования



**Приложение В**

# Cloud Computing

Студент:

| Группа | ФИО | Подпись | Дата |
|---|---|---|---|
| 8ИМ92 | Смирнов Павел Олегович | | |

Руководитель ВКР:

| Должность | ФИО | Ученная степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент отделения ИТ | Мирошниченко Евгений Александрович | К.т.н. | | |

Консультант-лингвист отделения иностранных языков ШБИП:

| Должность | ФИО | Ученная степень, звание | Подпись | Дата |
|---|---|---|---|---|
| Доцент ОИЯ | Коротченко Татьяна Валериевна | К.ф.н | | |

## 1. CLOUD COMPUTING

Prior to analyzing the topic related to tracking the dependencies of project assemblies on external libraries/modules used in them, it is necessary to understand which environment and what services the environment is working with.

All the resources which are used in this course project are provided by the cloud platform and cloud computing.

To carry out more deep investigation of the issues related to cloud computing, it is required to answer the basic questions presented in [1]. It would give a deeper insight into this phenomenon, i.e.:

- where the applications are located;
- key features of cloud computing;
- services provided;
- limits of manageability.

### 1.1 Where are the applications located?

When it comes to cloud computing, the issue concerning the deployment of the applications is of great significance. Today, three basic layout models are distinguished:

- customer's infrastructure;
- hosting company;
- cloud.

Deployment in a customer's infrastructure (on premises). This model is considered to be one of the most widely-applied. It has been already used for rather a long period of time. Hosting applications on the local infrastructure are accompanied with great investments at the first stage, precisely, these investments concern the hardware resources, software and network infrastructure.

Providing in a hosting company (hosting). This application delivery model, known at the time as the Application Service Provider (ASP) - SaaS or just "hosting" - was introduced a few years ago. Today it can be said that this model is one of the most widely used strategies for reducing infrastructure costs.

This includes the rental of the hardware platform, the software, the associated infrastructure and the staff who maintain them. This model offers less control over infrastructure, hardware and software and relies on the payment of a fixed number of resources, which usually involves a process.

Cloud deployment. This model was recently developed. When using leased hardware and software resources, payment is required, which significantly reduces acquisition costs and moves from capital investments to operating costs. In addition, no control over infrastructure or hardware is required.

It is obvious that all of these models have both advantages and disadvantages. However, it can be said that pay-as-you-go is one of the main parameters. This is the parameter that cloud computing has.

Cloud computing is an approach to hosting, serving, and consuming applications and computing resources. These applications and resources are now available on the Internet as services used on various platforms and devices [2]. Payment for these services is based on their actual usage.

### 1.2 Key features of cloud computing

The key features of cloud computing are [3]:

**Scalability**

A scalable application allows consumers to maintain a large load, by increasing a number of simultaneously running instances. Typically, standard hardware is used to run multiple instances simultaneously; precisely, this fact contributes to cutting down the total rent costs and infrastructure maintenance.

**Elasticity**

A measure that denotes the ability to adapt the workload by providing and disabling computing resources for automation tools.

**Multitenancy**

A mode that allows a single instance of an application to work simultaneously with multiple consumers at the same time.

**Payment for use**

Resources are scaled in such a way that the amount of resources provided to the consumer varies from directly to the current required capabilities. Scaling can be either horizontal, i.e. the expansion of the number of servers, or vertical, when the capacity of existing devices increases.

**Self-service**

The user himself is able to maintain and monitor the state of the systems and manage the equipment.

As you can see, these factors allow the user to reduce costs and increase productivity in the shortest possible time. Cloud technologies are increasingly integrated into new systems in such a way that simply ignoring them will only go to the loss of the enterprise.

### 1.3 Cloud computing and services they provide

Cloud computing and the services it provides (such as computing power or storage) can be compared to utilities. Just as the consumption of water and electricity changes in hot or cold weather, the consumption of services provided by cloud platforms can increase or decrease with increasing or decreasing load.

The similarity of services and utilities has several aspects. First, in both cases, consumers only pay for the actual disposal. Secondly, these two leased resources, the providers of these services, ensure their availability in the form of leased "resources", while reserving the problems in the creation and maintenance of the infrastructure. Third, when concluding a contract with an organization, expect the availability of certain resources and the organization - the timely payment of your rent.

Which services are most often provided by cloud platforms? Application hosting, data storage and computing are the most common scenarios for using cloud platforms. The following main services provided by cloud platforms can be distinguished [4].

### 1.4 Cloud services and the limits of manageability

When talking about different types of cloud services - software, platforms, and infrastructure as a service - you need to be aware of the limits of manageability. Compared to traditional deployment models, you can manage this in your own infrastructure when you switch to a cloud platform [5]. The differences in the controllability limits are shown in Figure 1.

IaaS services are relevant for startups and small businesses as well as large companies. Cloud services are an alternative to buying devices and building local infrastructure. As demand increases, companies are forced to implement new services and applications, which is made easier by the flexibility of cloud services. Put simply, migrating to IaaS saves time and money.

PaaS may be preferable for companies with an existing IT infrastructure. Customers need their own IT staff to use and configure the PaaS platform software. In return, however, the company has better control over the development process and the flexibility to deliver the finished application to customers.

Using SaaS services is beneficial for companies that cannot purchase on-premise solutions. Large companies can use this model for short-term projects that require quick, easy, and inexpensive solutions.


Figure 1 – Limits of manageability

Figure 1 shows that when you deploy your own infrastructure, you manage everything from network resources to running applications.

### 1.5 Microsoft Azure Platform

There are many cloud-computing platforms on the market today.

In order to find the platform or the provider that meets all requirements, it is important to clearly develop and define these requirements for the cloud and to carry out tests for all possible platforms. This is often the best way to understand if a solution is appropriate or if you should try building your own based on open platforms.

In the course project is used Azure platform from Microsoft.

In 2010, Microsoft began offering IT infrastructure services to companies in the form of web services, that is, according to the model that is now known as "cloud computing".

Today, Azure offers a highly reliable, scalable and cost-effective infrastructure platform in the cloud, which is used by hundreds of thousands of companies in 190 countries around the world.

#### 1.5.1 Platform Overview

The Azure cloud platform includes over 200 cloud products and services that you can use to create new solutions for the challenges of today and tomorrow. Build, run and manage applications in multiple clouds, on-premises and on the edge, with your preferred tools and platforms [6].

The Azure platform provides a set of services that are similar to those used by developers of «traditional» applications:

- **Computing services.** They are containers for applications with support for modern development technologies, including .NET, Java, PHP, Python, Ruby on Rails, and native code.
- **Data storage services.** A scalable distributed storage system that supports a number of storage models, including table structures, binary objects, asynchronous message queues, traditional file systems, and content distribution networks (CDNs).

– **Communication services.** Available via the cloud service bus and can be used as a messaging tool or connection broker with other cloud services or services hosted by customers.

– **Security services.** Policy-based access control services that support federation mechanisms and allow integration with existing identity management systems.

– **Application services.** Components and services those can be used to develop cloud applications and application services.

### 1.5.2 Cloud Platform Components

Azure Azure includes a great number of cloud services that can be used in combinations that meet all kinds of business requirements and organizational tasks (Figure 2).

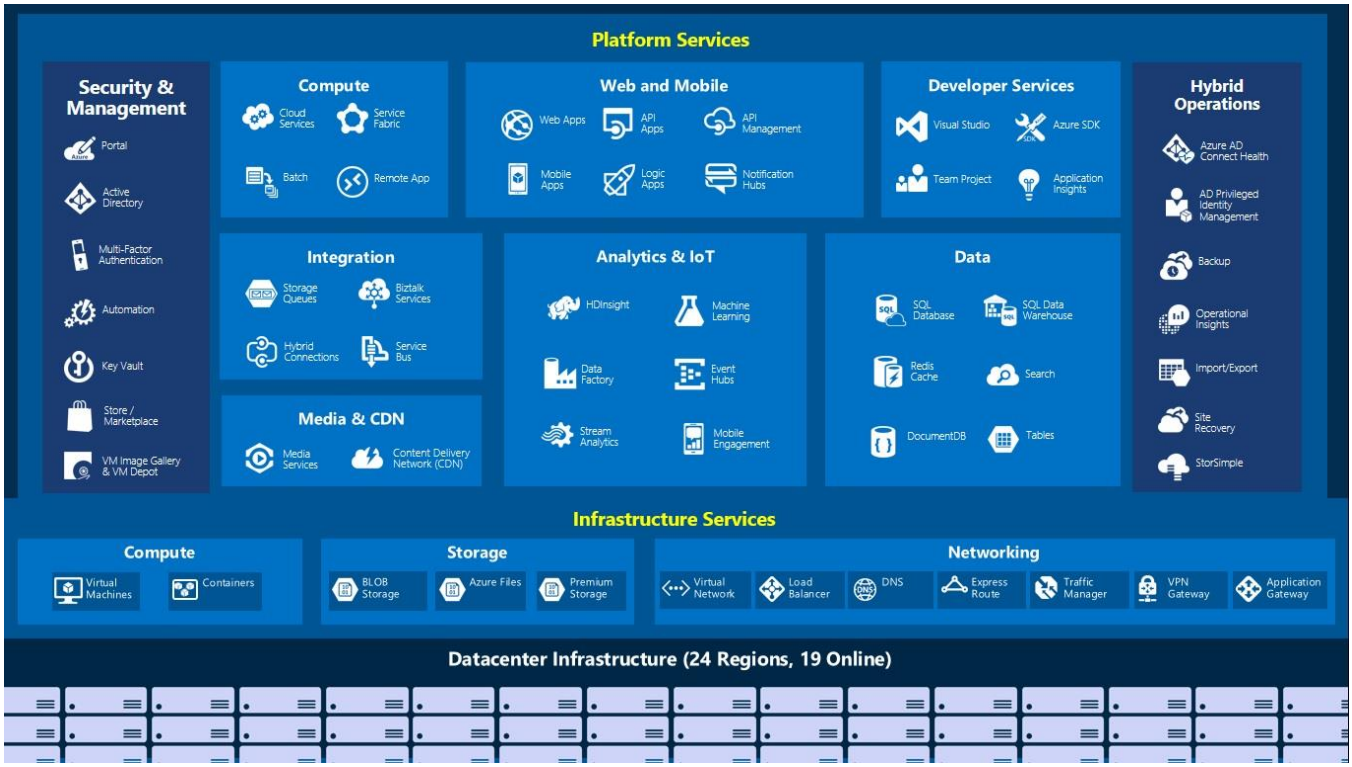



Figure 2 – List of Azure Cloud Services

There are the main Azure services that were actively used during the work [7].

#### 1.5.2.1 Computing

- The Virtual Machines service allows users to run general-purpose virtual machines running Microsoft Windows, Linux, and pre-configured images of popular service packages.
- The App service allows developers to publish and manage websites.
- The website hosting service allows developers to create websites using .NET, PHP, Node.js or Python and deploy them using FTP, Git, Mercurial, Team Foundation Server or upload via a custom portal.

#### 1.5.2.2 Data Management

- Storage Services provides REST and SDK APIs for storing and accessing data in the cloud.
- SQL Database, designed to create, scale, and extend applications in the cloud using Microsoft SQL Server technology.

#### 1.5.2.3 Working in the network

- Load Balancer allows you to scale applications and ensure the availability of services. Also supports incoming and outgoing scripts, provides low latency and high throughput.An application gateway is a web traffic load balancer designed to manage web application traffic.
- An application gateway is a web traffic load balancer designed to manage web application traffic.

### 1.6 Azure DevOps

Integration of cloud technologies has fundamentally changed the way creating, deploying and operating of applications. By implementing the DevOps methodology, teams gain additional opportunities to improve their work practices and customer service.

Azure DevOps provides services for developers to support teams to plan work, collaborate on code development, and build and deploy applications. [8]

### 1.6.1 DevOps methodology

DevOps is a methodology for active interaction of development specialists with information technology service specialists and mutual integration of their work processes into each other to ensure product quality [9].

DevOps affects the application lifecycle at all stages — from planning and development to delivery and operation. Each stage depends on the others, but the stages themselves do not depend on the roles performed by the company's employees. In a fully implemented DevOps culture, each role is more or less involved at each stage.

In addition to implementing the DevOps culture, teams implement the DevOps approach by applying specific techniques throughout the application lifecycle. Some of these techniques help speed up, automate, and improve the execution of a particular stage. Others cover multiple stages, helping teams create consistent processes that help increase productivity. It is important to understand that using the DevOps methodology. These methods are mostly automated, which results in high productivity in a team, since it skips routine steps that constantly occur during the development process.

The main methods used in this methodology are listed below:

#### 1.6.1.1 Continuous Integration

This is the practice of software development, which consists of merging working copies into a common main development branch several times a day and performing frequent automated builds of the project to identify and solve integration problems as soon as possible.

#### 1.6.1.2 Automated Testing

This is a software verification process in which the main functions and steps of the test, such as starting, initializing, executing, analyzing, and delivering the result, are performed automatically using automated testing tools.

#### 1.6.1.3 Continuous Deployment

Combining continuous integration and continuous delivery. This is the next step after a successful build and successful completion of the automated tests.

If you are confident in your tests, their set and coverage, when they are successfully completed, an automatic installation is launched on the appropriate environment, test or product.

#### 1.6.1.4 Infrastructure as Code

This is a process of managing and preparing computing infrastructure (processes, physical servers, virtual servers, etc.) and configuring them through machine-processed definition files, rather than using physical hardware configuration or configuration tools.

#### 1.6.1.5 Application Performance Monitoring

This is the monitoring and management of software performance and availability. APM strives to identify and diagnose performance issues in a comprehensive manner to maintain the expected level of service.

#### 1.6.1.6 Load Testing

A subspecies of performance testing, collecting metrics and determining the performance and response time of a software and hardware system or device in response to an external request in order to establish compliance with the requirements for a system (device).

#### 1.6.1.7 Release Management

Release management consists in defining the formal criteria for whether the build is ready to be installed on the appropriate environment. An example of the criteria by which the build is ready and running:

- DEV environment – build passed without errors.
- STAGE environment – build was installed on DEV environment and unit tests were successful.
- PROD environment – build has been tested on STAGE environment, there are no more than 5% minor errors, no major errors.

### 1.6.2 Pipelines

To maximize the effectiveness of the development team and accelerate market entry, it is important to plan and implement the deployment of the DevOps CI and CD pipelines.

The CI/CD pipeline describes an approach that simplifies the process of combining newly written code with existing code. This concept also allows you to run different types of tests at each stage and complete testing by running and deploying newly written code in an already tested version of software that end users can see.

The **DevOps CI/CD pipeline** is a necessary attribute of software development that includes any of the Agile approaches. This is where the issue of testing automation is concerned, which provides the fastest possible feedback between individual teams of specialists working on the project [10].

As for the main stages of the DevOps pipeline process, there are only five of them:

1. configuring the CD/CI pipeline DevOps infrastructure (framework) ;
2. integration with the source code management tool;
3. connecting the creating automation tool;
4. server-side software deployment;
5. testing the program code.

Based on the above, the ultimate benefits of deploying a DevOps pipeline include:

- faster software development lifecycle;
- implementation of effective quality assurance solutions;
- automation of software creation and release;
- configure the smooth operation of software.

In fact, the pipeline describes a set of methods for automating the deployment of applications in various environments, which allows you to make releases more frequent, reduce the number of hard-to-fix errors and associated downtime, and speed up the work of various groups.

itechx

itechx GmbH | Zentrale Saarbrücken | Innovationsring 20 | 66115 Saarbrücken

Herrn
Pavel Smirnov

Im Hause

Saarbrücken, 14. Juni 2021

**Bescheinigung / Confirmation**

Sehr geehrter Herr Smirnov,

gerne bestätigen wir Ihnen, dass Sie seit dem 01.01.2021 bis voraussichtlich zum 30.09.2021 bei uns beschäftigt sind.

Sie haben im Rahmen des Projekts PipelineTrigger selbstständig die technische Umsetzung übernommen und die Lösung mit C# und ASP.NET Core programmiert. Zur Versionsverwaltung wurde git verwendet. Darüber hinaus haben Sie das automatisierte Deployment (CI/CD) der Anwendung mit Azure DevOps Pipelines und Azure ARM Templates selbstständig umgesetzt.

Der Betrieb der Anwendung in Azure läuft seit Februar 2021 automatisiert und problemlos. Änderungen am Quellcode werden durch die konfigurierten Pipeline-Prozesse automatisiert in Produktion übertragen.

Der Service PipelineTrigger ist fester Bestandteil der von uns verwendeten Entwicklungsplattform Azure DevOps und wird produktiv zur Erstellung aller itechx Produkte eingesetzt.

---

Dear Mr Smirnov

We are pleased to confirm that you have been employed from January 1st, 2021 until probably September 30, 2021.

In the scope of the PipelineTrigger project you independently took over the technical implementation and programmed the solution using C# and ASP.NET Core. Git was used for version management. In addition, you independently implemented the automated deployment (CI/CD) of the application using Azure DevOps Pipelines and Azure ARM Templates.

The operation of the application in Azure has been automated running smoothly since February 2021. Changes to the source code are automatically transferred to production through the pipeline process configured.

The PipelineTrigger service is a standard component of the Azure DevOps development platform we use and is used productively to create all itechx products.

Mit freundlichen Grüßen / Best regards

Astrid Grahn

i.A. Astrid Grahn
Assistentin der Geschäftsführung, itechx GmbH