**Описание жизненного цикла программы**

# «Интеграционная платформа для организации продаж железнодорожных билетов и сопутствующих услуг со встроенным биллингом»

**Оглавление**

**Перечень сокращений**

| Термин | Определение |
|---|---|
| База данных (БД) | Совокупность данных, организованных в соответствии с концептуальной схемой, описывающей характеристики этих данных и связи между соответствующими им объектами, поддерживающая одну или несколько предметных областей |
| Доступ к информации (Доступ) | Ознакомление с информацией, ее обработка, в частности, копирование, модификация или уничтожение информации |
| Пользователь | Лицо, сотрудник Заказчика или организаций-агентов, участвующее в функционировании Системы или использующее результаты ее функционирования |
| ПО | Программное обеспечение |
| ПК | Персональный компьютер |
| ПАУ | Проектно-аналитическое управление |
| Система | Интеграционная платформа для организации продаж ЖД билетов и сопутствующих услуг организациями агентами со встроенным биллингом и порталом самообслуживания сотрудников организаций агентов |
| ФЗ | Функциональный заказчик |
| ФЭУ | Финансово-экономическое управление |
| ЗП | Заработная плата |
| КТУ | Коэффициент трудового участия |
| ТЗ | Техническое задание |
| СТП | Служба технической поддержки |
| UI | Графический интерфейс пользователя. |
| API | Программный интерфейс приложения. Описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. |

## 1 Общее описание

Процесс жизненного цикла выглядит следующим образом:

Приобретение -> поставка-> разработка -> функционирование -> сопровождение.

Процесс приобретения определяет действия предприятия – покупателя информационной системы, программного продукта или службы программного обеспечения.

Процесс поставки определяет действия предприятия-поставщика по снабжению покупателя информационной системой, программным продуктом или службы программного обеспечения.

Процесс разработки определяет действия предприятия-разработчика, который разрабатывает принципы построения программного изделия и собственно программный продукт.

Процесс функционирования определяет действия предприятия-оператора, обслуживающего систему в целом. Сюда входят консультация пользователей, получение обратной связи и т.д.

Процесс сопровождения определяет действия персонала, обеспечивающего сопровождение программного продукта, т.е. управление модификацией программного продукта, поддержку текущего состояния и функциональной пригодности, установку и удаление.

Также существует информация по вспомогательным процессам:

1) процесс решения проблем;

2) процесс документирования;

3) процесс управления конфигурацией;

4) процесс обеспечения качества; (устранение неисправностей, выявленных в ходе эксплуатации программного обеспечения)

5) процесс верификации;

6) процесс аттестации;

7) процесс совместной оценки;

8) процесс аудита.

## 2 Порядок предоставления информации по учёту рабочего времени производственного персонала ООО «ОНЭЛИЯ»

### 2.1 Схема реализации функционала

Процесс создания, разработки задач, что в последствии дает информацию о трудозатратах производственного персонала, выглядит следующим образом:

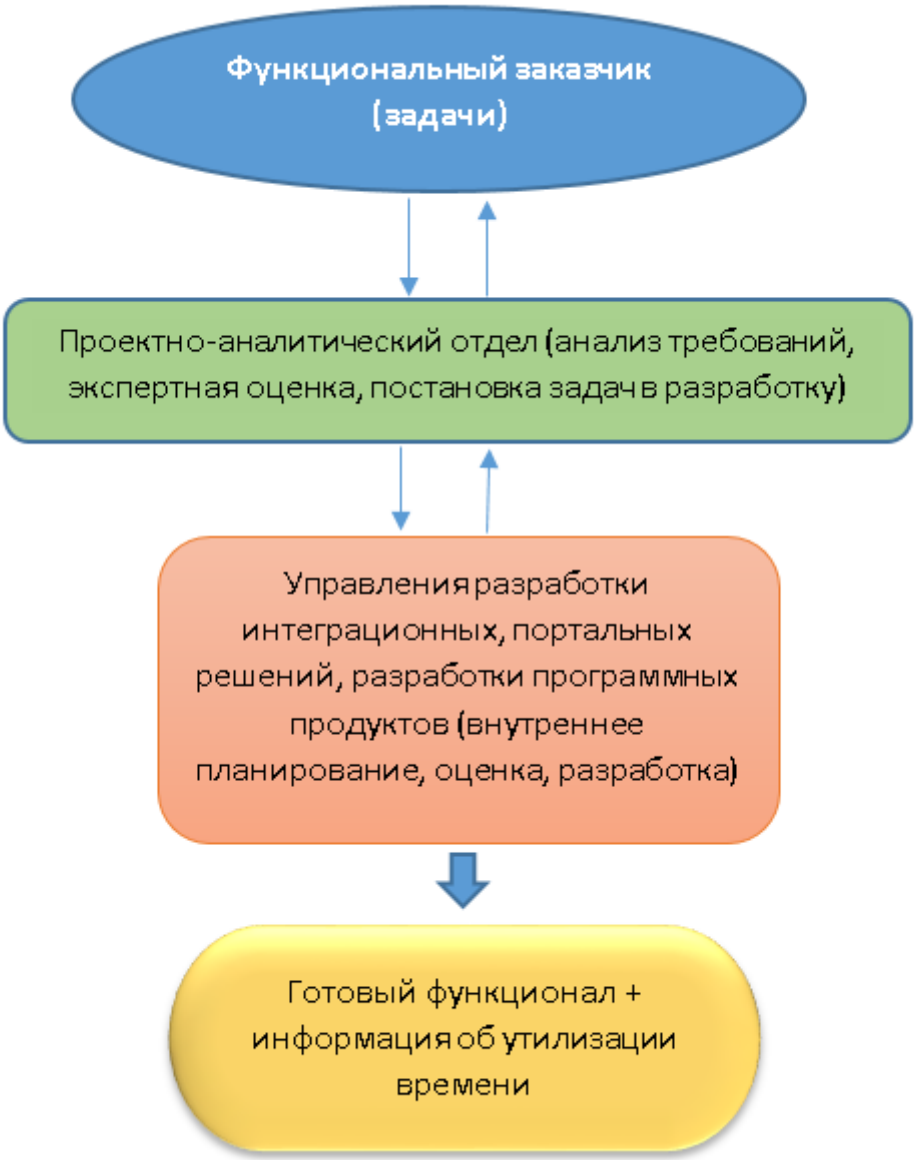



Рисунок 1 – Схема реализации функционала

### 2.2 Проработка и постановка задачи аналитиками

Руководитель проектно-аналитического управления (ПАУ) получает задачи от функционального заказчика (ФЗ) и назначает для каждой задачи аналитика на проработку. Аналитик проводит первичный анализ требований, на предмет того, что необходимо для

выполнения задачи. Далее происходит первый этап планирования функционала для дальнейшей аллокации по командам разработки с участием руководителей управлений и Директора по разработке программного обеспечения. При необходимости, если данных недостаточно, аналитик ПАУ уточняет требования у функционального заказчика (ФЗ). После уточнения всех требований у ФЗ (если этот шаг присутствовал) аналитик обращается в соответствующие команды разработки для получения экспертной оценки по трудозатратам на разработку задачи и определению подхода к разработке. Далее аналитик ПАУ согласовывает концептуальное решение задачи совместно с ФЗ.

В случае несогласования решения со стороны ФЗ, аналитик ПАУ повторно обращается в соответствующие команды разработки для переоценки задачи.

После согласования принятого решения по реализации задачи с ФЗ, аналитик ПАУ производит оценку данной задачи:

- проработка необходимых изменений в существующих программных продуктах;
- декомпозиция задачи;
- структурирование материала до уровня технического задания;
- постановка в трекерах;
- оценка на написание сопутствующей технической документации (при необходимости).

На аналитику закладывается 30 - 50% от итоговой оценки задачи на разработку и зависит от сложности задачи.

После валидации данной итоговой оценки задачи руководителем ПАУ, данная оценка согласуется с Техническим Директором, и, в случае подтверждения, направляется в сторону ФЗ для согласования. Задача уходит в разработку.

Руководитель ПАУ, владея информацией о занятости сотрудников в проектно-аналитическом управлении в течение месяца в различных проектах и задачах, проставляет процентные соотношения занятости по проектам в файле «Коэффициент трудового участия».

Примеры постановки и проработки задачи аналитиком в программе jira:

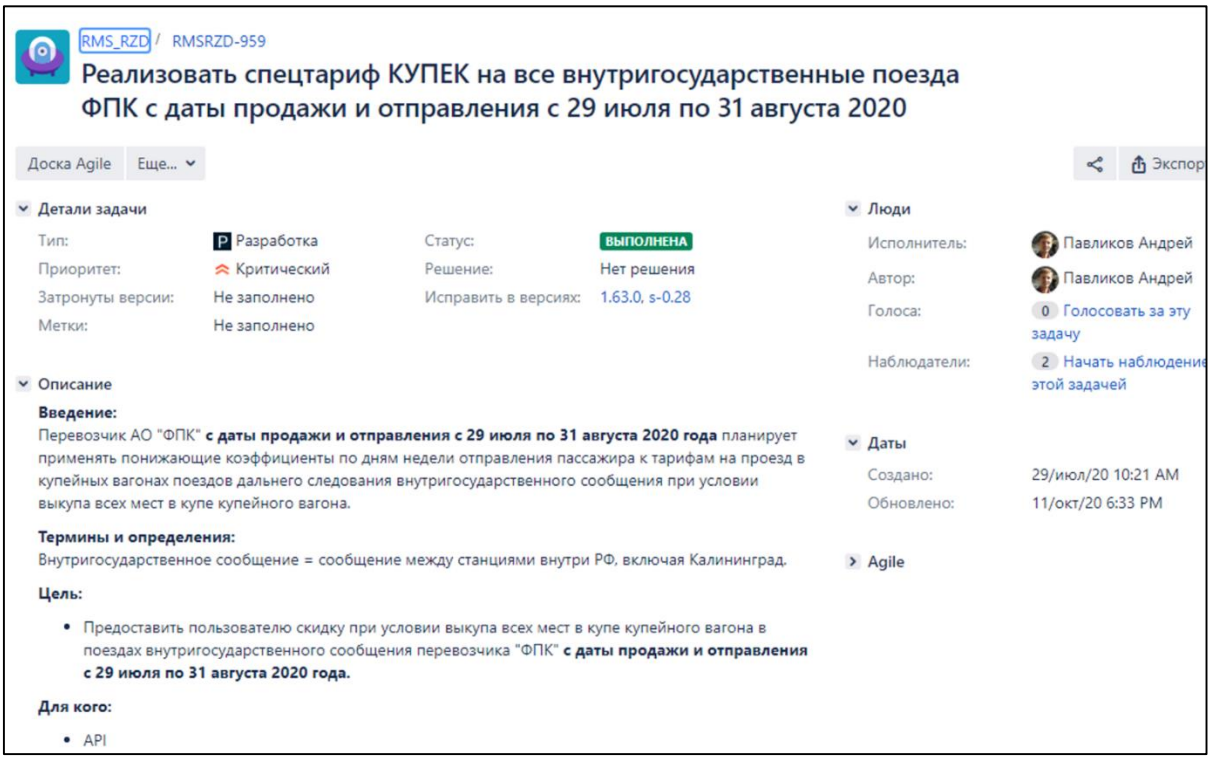



Рисунок 2 – Проработка задачи аналитиком

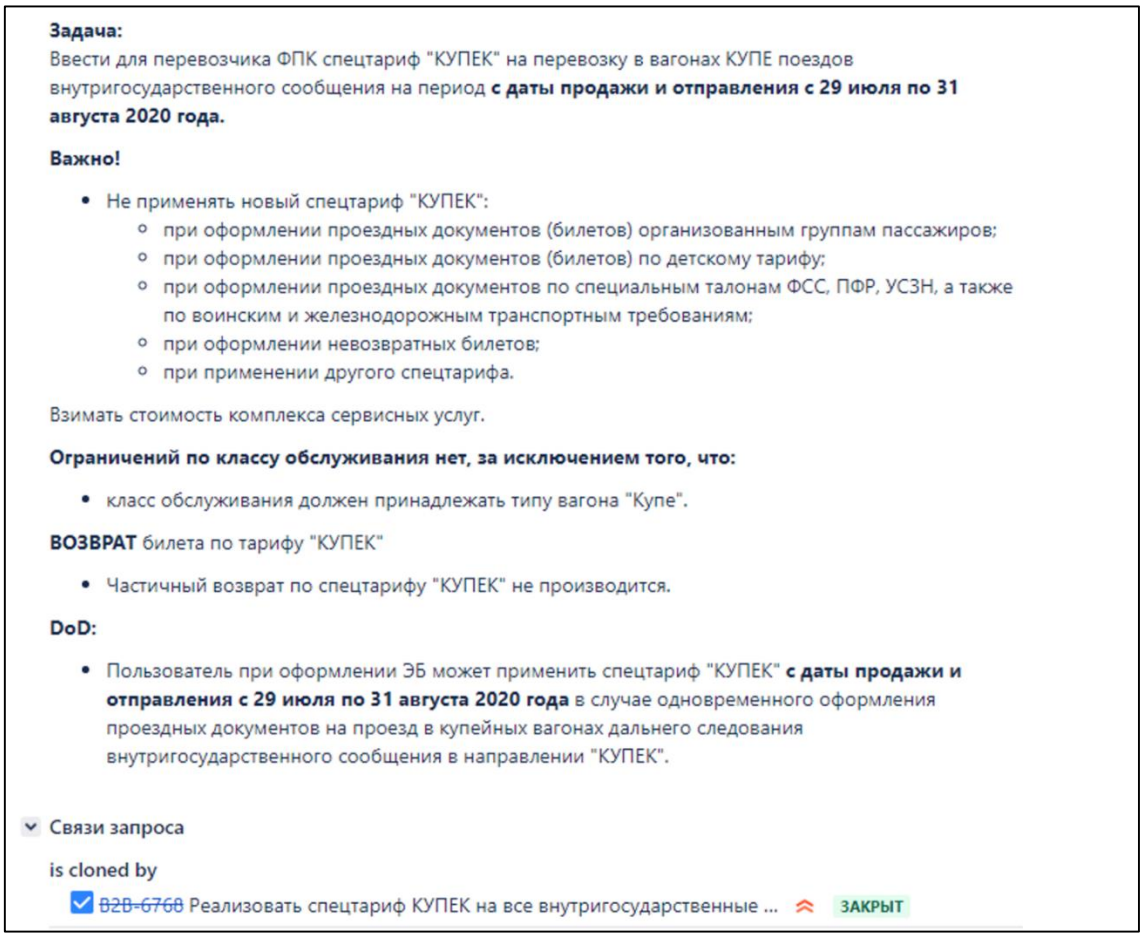



Рисунок 3 – Описание задачи

## 2.3 Проработка и реализация задач командами разработки

После проработки задач аналитиками, задачи распределяются в соответствующие управления разработки Директором по разработке программного обеспечения.

Перед началом реализации, все задачи проходят внутреннее планирование в командах разработки, где происходит оценка задачи разработчиками и тестировщиками, ее декомпозиция при необходимости.

Руководитель соответствующего структурного подразделения назначает исполнителей на задачи: разработчиков и тестировщиков.

Задачи после постановки попадают в бэклог прокета команды разработки. Там задачи приоритизируются совместно с менеджером продукта.

Разработка происходит отрезками по четыре недели. Каждый из отрезков планируется по трудозатратам и приоритетности задач. После окончания проходит тестирование и формируется сборка для установки на продуктив.

Каждые четыре недели задачи проходят планирование в релиз. Предварительно они изучаются командой, проектируются и составляется план реализации. Все задачи оцениваются и берутся в работу исходя из приоритетов и емкости команды на четыре недели.

После планирования и начала итерации задачи распределяются на разработчиков. Для выполнения работы по задаче разработчик формирует отдельную фиче ветку в репозитории от последней стабильной версии. В рамках фиче ветки проходит разработка нового функционала.

После завершения разработчик пишет краткое описание в задаче того, как была выполнена задача и на что следует обратить внимание тестировщику и переводит задачу на кодревью.

В рамках кодревью другой разработчик из команды или старший разработчик проводит проверку согласно принятым нормам написания кода в команде, логику и правильность алгоритма, проверяет наличие тестов на написанный функционал. По итогу проверки задача может быть отправлена на доработку.

После завершения кодревью задача переходит в тестирование. Тестировщик проводит все необходимые проверки согласно конкретной задаче. Пишет сценарии тестирования и регистрирует факты проверки. Если задача не проходит тестирование она может быть отправлена на доработку.

После успешного прохождения тестирования код задачи из фиче ветки вносится в ветку релиза, для формирования релиза.

После того как все задачи, запланированные на итерацию перенесены в ветку релиза, начинается создание сборки под релиз. Запускаются автотесты и по успешному их прохождению

формируется сборка. Сборка разворачивается на тестовом полигоне для прохождения регрессионного тестирования.

Согласно регресс плану тестирования команда тестирования выполняет полную проверку решения перед его установкой на пред продуктовый стенд.

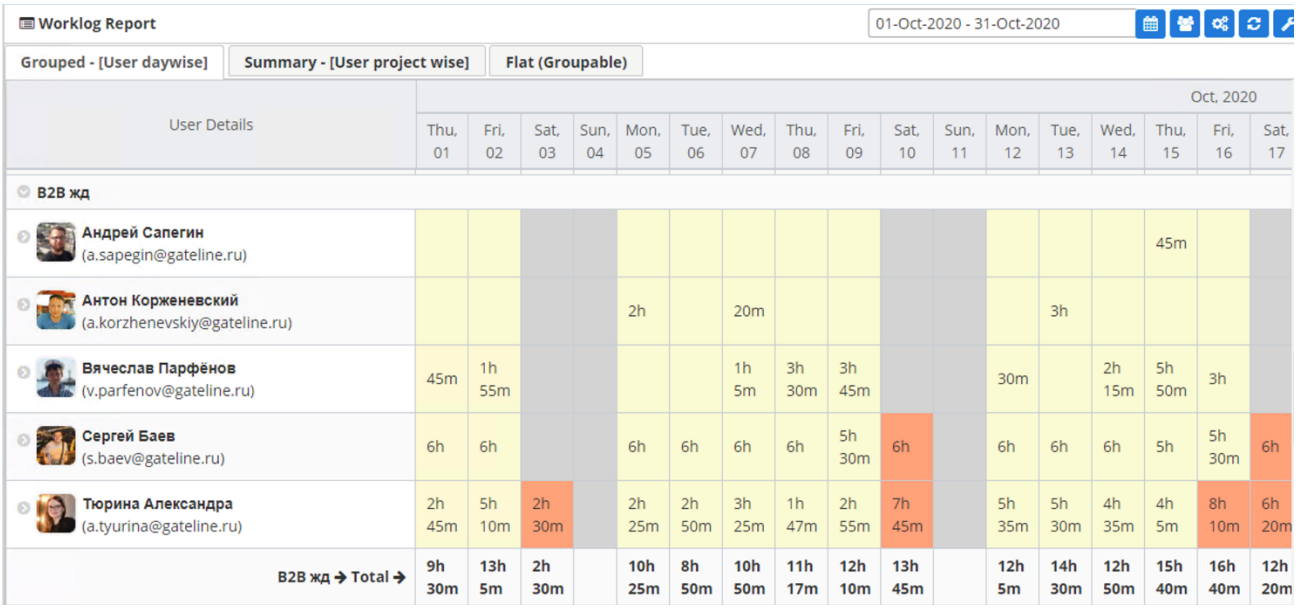
Если регрессионное тестирование выявило проблемы или ошибки, разработчик выполняет внесение необходимых изменений в код и сборка формируется заново.

По итогу регрессионного тестирование руководитель тестирования пишет заключение и принимается решение по установке релиза на пред продуктовый стенд.

На пред продуктовый стенд проходит короткое смоук тестирование и, если все работает корректно, то стенд переключатся в продуктив.

## 2.4 Выгрузка отчета о трудозатратах производственного персонала

На основании указанных данных происходит выгрузка отчета о трудозатратах в Jira (изображение ниже):



Worklog Report — 01-Oct-2020 - 31-Oct-2020

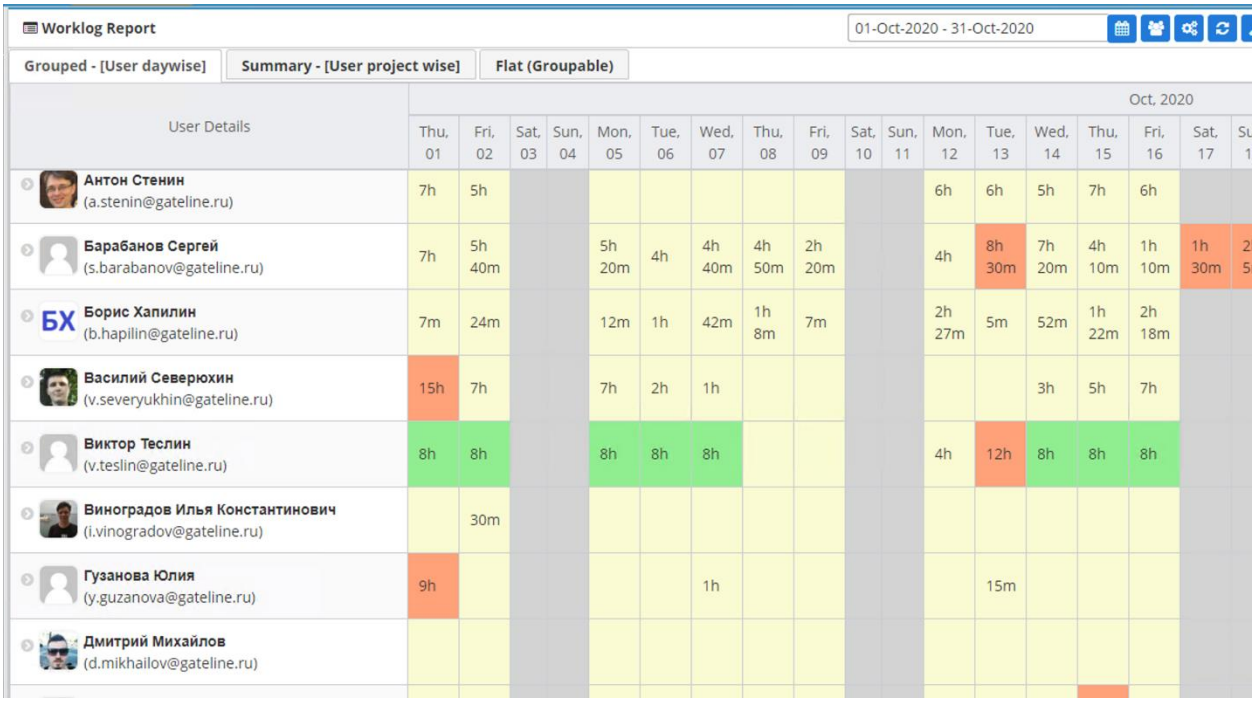
Grouped - [User daywise] | Summary - [User project wise] | Flat (Groupable)

| User Details | Oct, 2020 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Thu, 01 | Fri, 02 | Sat, 03 | Sun, 04 | Mon, 05 | Tue, 06 | Wed, 07 | Thu, 08 | Fri, 09 | Sat, 10 | Sun, 11 | Mon, 12 | Tue, 13 | Wed, 14 | Thu, 15 | Fri, 16 | Sat, 17 |
| **B2B жд** | | | | | | | | | | | | | | | | | |
| **Андрей Сапегин** (a.sapegin@gateline.ru) | | | | | | | | | | | | | | | 45m | | |
| **Антон Корженевский** (a.korzhenevskiy@gateline.ru) | | | | | 2h | | 20m | | | | | | 3h | | | | |
| **Вячеслав Парфёнов** (v.parfenov@gateline.ru) | 45m | 1h 55m | | | | | 1h 5m | 3h 30m | 3h 45m | | | 30m | | 2h 15m | 5h 50m | 3h | |
| **Сергей Баев** (s.baev@gateline.ru) | 6h | 6h | | | 6h | 6h | 6h | 6h | 5h 30m | 6h | | 6h | 6h | 6h | 5h | 5h 30m | 6h |
| **Тюрина Александра** (a.tyurina@gateline.ru) | 2h 45m | 5h 10m | 2h 30m | | 2h 25m | 2h 50m | 3h 25m | 1h 47m | 2h 55m | 7h 45m | | 5h 35m | 5h 30m | 4h 35m | 4h 5m | 8h 10m | 6h 20m |
| **B2B жд → Total →** | **9h 30m** | **13h 5m** | **2h 30m** | | **10h 25m** | **8h 50m** | **10h 50m** | **11h 17m** | **12h 10m** | **13h 45m** | | **12h 5m** | **14h 30m** | **12h 50m** | **15h 40m** | **16h 40m** | **12h 20m** |

Рисунок 4 – Отчёт о трудозатратах в Jira

Так выглядит отчет утилизации рабочего времени персонала на кодирование, фильтр на отчет в программе по задачам «РЖД», Управление разработки интеграционных решений.

Ниже представлена информация оп подразделению Управление разработки портальных решений, где сотрудники также фиксируют время кодирования:



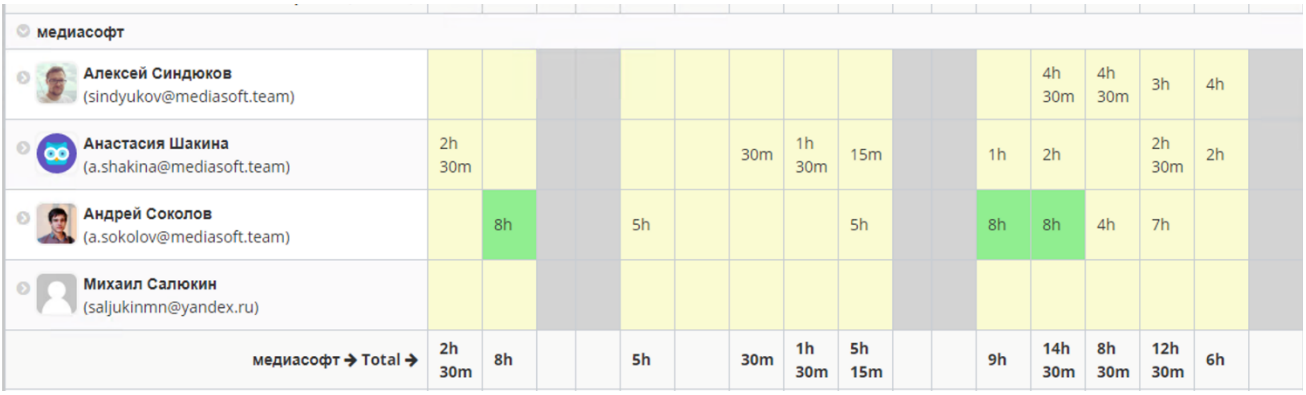
Worklog Report — 01-Oct-2020 - 31-Oct-2020

Grouped - [User daywise] | Summary - [User project wise] | Flat (Groupable)

| User Details | Oct, 2020 | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Thu, 01 | Fri, 02 | Sat, 03 | Sun, 04 | Mon, 05 | Tue, 06 | Wed, 07 | Thu, 08 | Fri, 09 | Sat, 10 | Sun, 11 | Mon, 12 | Tue, 13 | Wed, 14 | Thu, 15 | Fri, 16 | Sat, 17 |
| Антон Стенин (a.stenin@gateline.ru) | 7h | 5h | | | | | | | | | | 6h | 6h | 5h | 7h | 6h | |
| Барабанов Сергей (s.barabanov@gateline.ru) | 7h | 5h 40m | | | 5h 20m | 4h | 4h 40m | 4h 50m | 2h 20m | | | 4h | 8h 30m | 7h 20m | 4h 10m | 1h 10m | 1h 30m |
| БХ Борис Хапилин (b.hapilin@gateline.ru) | 7m | 24m | | | 12m | 1h | 42m | 1h 8m | 7m | | | 2h 27m | 5m | 52m | 1h 22m | 2h 18m | |
| Василий Северюхин (v.severyukhin@gateline.ru) | 15h | 7h | | | 7h | 2h | 1h | | | | | | | 3h | 5h | 7h | |
| Виктор Теслин (v.teslin@gateline.ru) | 8h | 8h | | | 8h | 8h | 8h | | | | | 4h | 12h | 8h | 8h | 8h | |
| Виноградов Илья Константинович (i.vinogradov@gateline.ru) | | 30m | | | | | | | | | | | | | | | |
| Гузанова Юлия (y.guzanova@gateline.ru) | 9h | | | | | | 1h | | | | | | 15m | | | | |
| Дмитрий Михайлов (d.mikhailov@gateline.ru) | | | | | | | | | | | | | | | | | |

Рисунок 5 – Информация оп подразделению Управление разработки портальных решений

Рассмотрим также пример фиксации времени кодирования внешними сотрудниками:



медиасофт

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Алексей Синдюков (sindyukov@mediasoft.team) | | | | | | | | | | | | | 4h 30m | 4h 30m | 3h | 4h |
| Анастасия Шакина (a.shakina@mediasoft.team) | 2h 30m | | | | | | 30m | 1h 30m | 15m | | | 1h | 2h | | 2h 30m | 2h |
| Андрей Соколов (a.sokolov@mediasoft.team) | | 8h | | | 5h | | | | 5h | | | 8h | 8h | 4h | 7h | |
| Михаил Салюкин (saljukinmn@yandex.ru) | | | | | | | | | | | | | | | | |
| **медиасофт → Total →** | **2h 30m** | **8h** | | | **5h** | | **30m** | **1h 30m** | **5h 15m** | | | **9h** | **14h 30m** | **8h 30m** | **12h 30m** | **6h** |

Рисунок 6 – Пример фиксации времени кодирования внешними сотрудниками

Для понимания разработчиков и аналитиков, в трекере введена метка «продукт», где сотрудники отмечают принадлежность каждой задачи к «продукту» на этапе постановки и разработки.

Файл «КТУ» содержит утилизацию рабочего времени производственного персонала по проектам. Каждый проект в файле «КТУ» может содержать несколько «продуктов». Ежемесячно, аналитик агрегирует информацию по утилизации рабочего времени производственного персонала по «продуктам», в составе проектов.

Таким образом, мы получаем ежемесячно информацию по учету рабочего времени производственного персонала, что в последующем дает возможность рассчитать %-ую загруженность производственного персонала для файла «КТУ» (коэффициент трудового участия).

| | | | | МДС | | | | B2B ж/д (для ИМ) |
|---|---|---|---|---|---|---|---|---|
| | | | | Инвестиционные проекты (разработка новых версий ПО) | | | МДС неисключительные права/ коммерция (в т. ч. ЭКСПЛУАТАЦИЯ и авторское сопровождение) | Авторское сопровождение и доработка для ИМ в части ж/д |
| | | | | Разработка МДС B2B (необходимо деление задач в jira для себя и ИМ) | Разработка B2C платформы для МДС (необходимо деление задач в jira для себя и ИМ) | МДС Проект 955/Гос часть (инвестиционный проект) | | |
| **Кузнецов Михаил Александрович** | Команда разработки 1 | 100,0% | | 35,4% | 2% | 15,0% | 3,3% | 20,6% |
| Баев Сергей Викторович | Команда разработки 1 (жд) | 100,0% | | 40,0% | 2% | | 3,3% | 17,3% |
| Парфенов Вячеслав Александрович | Команда разработки 1 (жд) | 100,0% | | 40,0% | 2% | | 3,3% | 17,3% |
| Корженевский Антон Александрович | Команда разработки 1 (жд) | 100,0% | | 40,0% | 2% | | 3,3% | 17,3% |
| Салегин Андрей Юрьевич | Команда разработки 1 (жд) | 100,0% | | 40,0% | 2% | | 3,3% | 17,3% |
| Тюрина Александра Николаевна | Команда разработки 1 (жд) | 100,0% | | 40,0% | 2% | | 3,3% | 17,3% |
| Волков Станислав Львович | Команда разработки 1 (авиа+аэро) | 100,0% | | 94,7% | 2% | | 3,3% | |
| Гусев Вадим Юрьевич | Команда разработки 1 (авиа+аэро) | 100,0% | | 94,7% | 2% | | 3,3% | |
| Кислов Данил Михайлович | Команда разработки 1 (авиа+аэро) | 100,0% | | 87,7% | 2% | 7% | 3,3% | |
| Трифонов Александр Владимирович | Команда разработки 1 (авиа+аэро) | 100,0% | | 94,7% | 2% | | 3,3% | |
| Шихалиев Руслан Сираджеддинович | Команда разработки 1 (авиа+аэро) | 100,0% | | 94,7% | 2% | | 3,3% | |
| Зайцев Максим Борисович | Команда разработки 1 (автоб.) | 100,0% | | 94,7% | 2% | | 3,3% | |
| Колосов Евгений Сергеевич | Команда разработки 1 (автоб.) | 100,0% | | 94,7% | 2% | | 3,3% | |
| Щукин Роман Леонидович | Команда разработки 1 (автоб.) | 100,0% | | 94,7% | 2% | | 3,3% | |
| Кочуров Сергей Александрович | Команда разработки 1 (автоб.) | 100,0% | | 94,7% | 2% | | 3,3% | |
| Борецкий Олег Игоревич | Команда разработки 1 | 100,0% | | 47,0% | 2% | | 3,3% | 33,2% |
| Диева Александра Витальевна | Команда разработки 1 | 100,0% | | 47,0% | 2% | | 3,3% | 33,2% |
| Дудин Вадим Андреевич | Команда разработки 1 | 100,0% | | 47,0% | 2% | | 3,3% | 33,2% |
| Зайцев Андрей Александрович | Команда разработки 1 | 100,0% | | 94,7% | 2% | | 3,3% | |

Рисунок 7 – Отчёт

В начале месяца, по итогам предыдущего месяца, полученные итоговые показатели утилизированных часов по проектам (программным продуктам) проходят процесс окончательного согласования с руководителями соответствующих групп разработки – второй уровень контроля и финально утверждаются Директором по разработке программного обеспечения – третий уровень контроля Таким образом мы получаем максимально достоверную информацию по учету рабочего времени производственного персонала ежемесячно. Данный файл передается в ФЭУ (финансово-экономическое управление) для обработки и аллокации ЗП и страховых взносов, как производственного персонала и АУП по проектам Компании.

Кроме указанных подразделений разработки и аналитики, в производственном блоке компании существуют также Управление эксплуатации ЦОД, Управление эксплуатации и контроля качества, Управление информационной безопасности, Техническое управление проектами. Утилизация рабочего времени сотрудников данных структурных подразделений определяется руководителями структурных подразделений на основании количества, объема задач и участия в проектах.

## 3 Регламент работы отдела тестирования

Настоящий регламент является частью организации процесса тестирования программного продукта B2B. В документе представлены обязательные процедуры, ограничения и правила организации тестирования в процессе разработки программного продукта. Данный регламент имеет статус стандарта и обязателен для выполнения всеми участниками тестирования программного обеспечения.

Целью данного документа является:

- организация процесса тестирования задач;
- организация контроля над процессом тестирования.

Тип взаимодействия – B2B. Совершать действия можно как через UI так и через API.

Продукт включает в себя:

- API;
- Бэкофис;
- ApiDocs.

Этапы тестирования задач в рамках релиза:

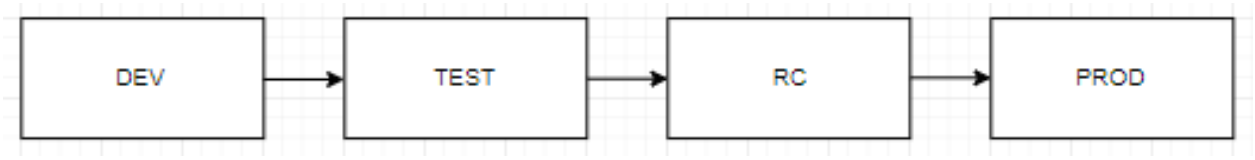



Рисунок 8 – Этапы тестирования

### 3.1 Тестирование на стенде Dev

URL для тестирования – http://dev.tripandfly.local/

Тестирование на стенде dev начинается после перевода задачи в статус QA и мерджа соответствующей ветки. В отдельных случаях, задачи могут быть реализованы сразу на dev.

Процесс обработки задачи на dev:

а) назначить роль QA на себя. Для этого в задаче в поле QA выбрать свою у.з.

б) проверить задачу, в соответствии с требованиями (прямыми в задаче/косвенными), при необходимости – провести регресс смежного функционала

в) в задаче создать новый комментарий с указанием: стенда тестирования (dev/feature/devsite/test и пр.), списка проведенных проверок, для каждой проверки прикрепить ссылку на скрин

г) в случае, если все проверки прошли успешно - перевести задачу в статус Done (кнопка QA Passed). В случае, если какие то из проверок не прошли – задача переводится в статус Reopened(кнопка QA Failed). В случае, если во время проверки изменились требования к задаче – перевод в статус TO DO(кнопка Claims update)

д) залогировать время (More→ LogWork: поле Time Spent)

## 3.2 Тестирование на стенде Test

URL для тестирования - http://test.tripandfly.local/

После того, как все задачи релиза были протестированы на стенде dev, все задачи сливают на следующий стенд - test. Отследить окончание заливки релиза можно в TeamCity в блоке "Deploy TEST test.tripandfly.local" с этого момента начинается вторая итерация тестирования релиза. На данном этапе происходит повторная проверка задач, назначенных на тестировщика (те, которые он проверял на dev стенде), прогон автотестов, а также регрессионное тестирование функционала системы (см. Процесс проведения регрессионного тестирования).

Последовательность действий для сотрудников:

Ручные тестировщики:

- проверка своих задач на стенде test;
- проведение регрессионного тестирования. Тест кейсы назначаются в TestRail https://onelya.testrail.io/ . Создается отдельная задача в Jira, в неё логируется время, затраченное на тестирование.

Автоматизаторы:

- запуск автотестов;
- занесение результатов выполнения тестов в чек-лист;
- проведение ручного регрессионного тестирования.

В случае, если при тестировании на данном этапе был найден дефект, необходимо его локализовать, а также выяснить, является ли он привнесенным.

Привнесенный дефект - дефект, который появился в результате выкладки какой-либо задачи текущего релиза. Определяется так: если найденная ошибка НЕ воспроизводится на стенде, на котором еще нет кода нового релиза - дефект считается привнесенным. Исправление найденного дефекта является приоритетным и производится в рамках текущего релиза. Исправлением дефекта занимается тот разработчик, чья задача привнесла дефект. Тестировщику в данном случае необходимо описать в комментариях к задаче найденный дефект (подробно и понятно, со скринами), сообщить разработчику и вернуть задачу на доработку.

(!) Если дефект был выявлен при прохождении автотестов, необходимо пройти зафейлиный кейс вручную и убедиться, в том, что он заваливается (либо это ошибка теста)

Не привнесенный дефект - дефект, который не является последствием заливки кода на стенд. Определяется так: если найденная ошибка воспроизводится на стенде, на котором еще нет кода нового релиза - дефект считается не привнесенным. В такой ситуации заводится новый дефект в Jira согласно требованиям Регистрация дефектов/ошибок в Jira В случае обнаружения критичной ошибки дефект исправляется в текущем релизе, в случае некритичной ошибки – в следующем.

(!) Для логирования времени на регрессионное тестирование, для каждого релиза создается задача в JIRA куда необходимо списывать время, затраченное на регрессионное тестирование

### 3.3 Тестирование на стенде RC

Тестирование на стенде RC не проводится (!)

### 3.4 Тестирование на стенде PROD

URL для тестирования - https://partner.fpc.ru/

Тестирование на стенде начинается после деплоя релиза на прод (этим занимаются админы и сообщают об успешном деплое). В связи с ограниченными правами на стенде (подтверждать покупку билетов нельзя, доступа к логам и Бэкофису нет), проводится проверка только критичных функций системы, а именно: Покупка билетов до части подтверждения покупки, оформление страховки ВЗР.

Важно: на странице подтверждения бронирования билета/страховки необходимо отменить заказ.

### 3.5 Процесс проведения регрессионного тестирования.

Проведение регрессионного тестирования традиционно проходит при деплое релиза на стенд test/testsite

#### 3.5.1 Создание задачи на регрессионное тестирование:

1.В JIRA в проекте B2B (B2B) необходимо создать новую задачу, в которой:

- Issue Type - Task
- SummaryRequired - название задачи с указанием версий релизов. *Например*, "Регрессионное тестирование релизов <версия релиза> и <версия релиза>"

– Component/s - QA

  – Fix Version/s - версия релиза, в рамках которого проводится регрессионное тестирование

  – Environment - стенды, на которых будет проводиться проверка кейсов

  – Description - в данном поле указываются ссылки на созданные TestPlan в TestRail

  – Code base - ALL

3.5.2 Создание плана тестирования в TestRail:

1 .Перейти в инструмент TestRail https://onelya.testrail.io/

2. В разделе "Milestones" создать Milestone в соответствующем проекте (GDS/SITE) с указанием названия, соответствующего версии релиза (например, 1.52.0)

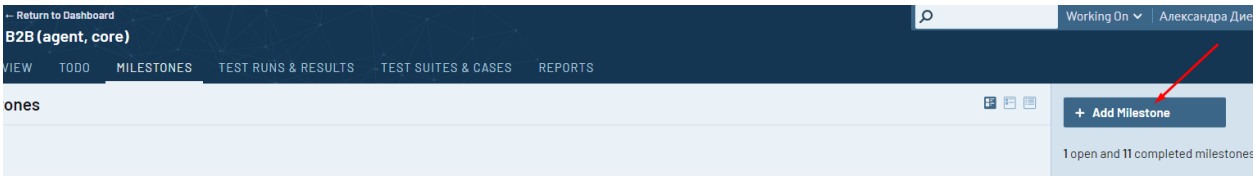



Рисунок 9 – Создание Milestone

3. В разделе "Test Runs & Results" создать новый TestPlan, указав в поле "Name" - "Тестирование релиза <версия релиза>", привязать соответствующий версии Milestone.

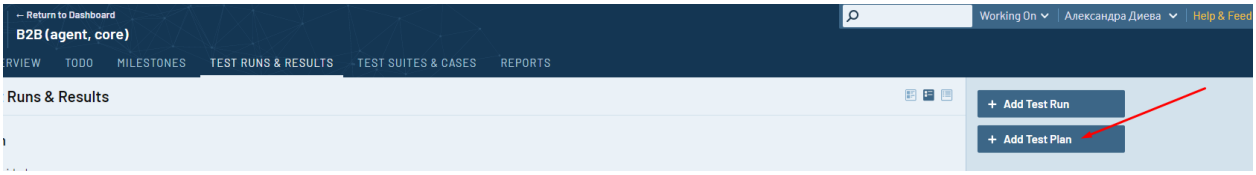



Рисунок 10 – Создание тест плана

4. Добавить в TestPlan наборы кейсов, выбрав на странице редактирования плана кнопку "Add Test Suite".

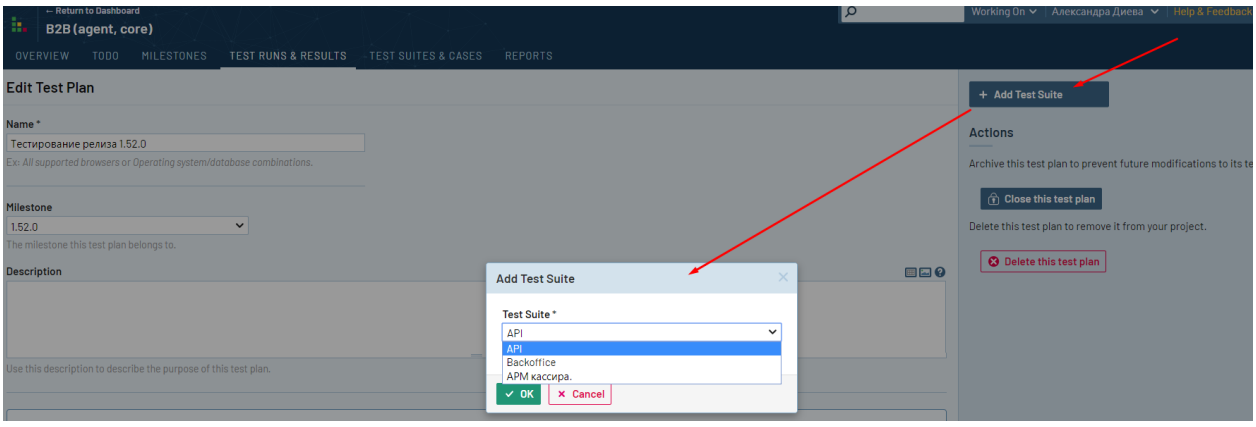



Рисунок 11 – Добавление тест плана

Опционально, для каждого TestSuite можно установить ответственного (все кейсы, входящие в набор, будут назначены на выбранного пользователя), добавить описание и конфигурации. По-умолчанию, при добавлении TestSuite добавляются все тесткейсы, входящие в набор. Изменить список кейсов можно по ссылке select cases (рекомендуется оставлять полный список кейсов)

3.5.3 Описание типов ТестКейсов в TestRail:

– Regression. Такой тип имеют тесткейсы, которые проверяются вручную сотрудниками отдела тестирования.

– Automated. Данный тип используется для обозначения кейсов, которые уже автоматизированы, вручную их проверять не нужно.

Тесткейсы с типом Automation не проходятся руками, запуск тестов осуществляется из TeamCity в проекте QA (Процесс запуска автотестов).

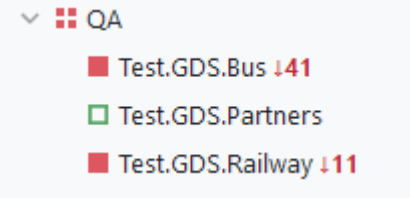



Рисунок 12 – Процесс запуска автотестов

После прохождения автотестов формируется отчёт в TestRail. Все сценарии со статусом Failed должны быть перепроверены руками (обычно ошибки связаны с работой смежных тестовых систем).

Тесткейсы с типом Regression распределяются по тестировщикам для выполнения.

3.5.4 Проверка кейсов из TestPlan:

1.в TestRail перейти в соответствующий TestRun. Откроется страница с списком кейсов. По-умолчанию список не отсортирован, фильтры не установлены. Для удобства просмотра можно отсортировать список/выставить фильтры. Например, для выборки всех не пройденных ручных кейсов, назначенных на текущего пользователя необходимо в поле Filter выбрать поля "Assigned To" → выбрать своего пользователя, "Status" → выбрать Untested, "Type" → выбрать Regression. (!) Тесткейсы не всегда назначены сразу на определенного сотрудника, распределяются порционно. При проверке задачи на стенде test, в случае если были затронуты какие-либо кейсы из регресса, статус для них проставляет сотрудник, проверивший их в рамках своей задачи. В случае если кейс ни на кого не назначен, его можно "забрать" себе. Как назначить тесткейс: активировать соответствующий кейсу чекбокс → нажать "Assign To" →

"Assign selected"→ в диалоговом окне в поле "Assign To" выбрать своего пользователя. Опционально можно указать комментарий

2. Перейти в карточку тесткейса. Откроется страница с описанием шагов и ожидаемого результата

3. Проверить тесткейс в соответствии с описанными шагами. (!) В случае обнаружения ошибок или неточностей в описании кейса, необходимо в задаче регрессионного тестирования (в Jira) в комментарии указать id кейса и неточность/ошибку. В дальнейшем по описанным замечаниям кейсы будут исправляться.

4. Проставить тесткейсу статус, отражающий результат проверки, нажав на кнопку "AddResult" Подробное описание статусов есть в данной статье в блоке "*Проставление статусов для пройденных кейсов TestRun"*

Проставление статусов для пройденных кейсов TestRun:

– Passed - кейс успешно пройден

– Blocked - выставляется в случае, когда тестирование функциональности недоступно по причине не связанной с проверяемым функционалом и не являющимся ошибкой функционала.

– Failed - выставляется в случае, если при тестировании кейса обнаружена ошибка, напрямую связанная с проверяемым функционалом. В данном случае необходимо в комментарии указать ошибку и в поле Defects указать номер задачи в JIRA

– Retest - выставляется в случае, когда необходимо по той или иной причине вернуться к проверке позже.

3.5.5 Завершение регрессионного тестирования.

TestPlan закрывается только при выполнении следующих условий:

– проверены все кейсы, входящие в план тестирования

– отсутствуют кейсы со статусом Retest

– привнесенные дефекты, выявленные в рамках тестирования исправлены в соответствующих задачах и разлиты по веткам. Исключение составляют минорные баги, исправление которых согласовано в следующем релизе.

1. Закрыть TestPlan. Для этого необходимо перейти в TestRail → страницу "Test Runs & Results" → перейти на страницу соответствующего TestPlan. В карточке testplan нажать на "Edit" и в правой части открывшейся страницы нажать на "Close this test plan"(При успешной операции отобразится сообщение "Successfully closed the test plan and the related test runs." в верхней части страницы)

2. Закрыть Milestone. Для этого необходимо перейти в TestRail → страницу "Milestones" → перейти на страницу соответствующего Milestone. В карточке milestone нажать на "Edit" и в нижней части открывшейся страницы активировать чекбокс "This milestone is completed" и сохранить изменения. При успешной операции отобразится сообщение "This milestone is completed".

3. В JIRA перевести задачу по регрессионному тестированию в статус Closed. Все участвующие в тестировании сотрудники логируют в задачу затраченное время через More - > LogWork.

## 4 Размещение инфраструктуры и персонала, требования к персоналу

### 4.1 Размещение инфраструктуры и персонала

Размещение инфраструктуры и персонала

1. Адрес фактического размещения инфраструктуры разработки: Москва, ул. Авиаконструктора Микояна, 12, БЦ "Линкор", корпус Б, 6 этаж, помещ. I, комн. 32,33,34,35,37,38.

2. Адрес размещения разработчиков: Москва, ул. Авиаконструктора Микояна, 12, БЦ "Линкор", корпус Б, 6 этаж, помещ. I, комн. 32,33,34,35,37,38.

3. Адрес размещения службы поддержки: Москва, ул. Авиаконструктора Микояна, 12, БЦ "Линкор", корпус Б, 6 этаж, помещ. I, комн. 32,33,34,35,37,38.

### 4.2 Требования к персоналу

Количества персонала:

- разработка – 3 человека;
- аналитик – 3 человека;
- специалист технической поддержки – 2 человека.

Требования к разработке:

1. Требования к опыту:

- веб-разработка не менее 3-х лет;
- отличные знания C#, .Net Core, .Net Framework 4.6, REST;
- знание SQL;
- понимание принципов TDD, владение инструментами Unit-тестирования;
- знание принципов ООП, шаблонов проектирования, SOLID;
- опыт использования систем контроля версий (Git и др.).

2. Приветствуются:

- знание и опыт работы с Web API, Entity Framework;
- опыт работы в команде по методологии Agile;
- опыт работы с PostgreSQL.

Требования к аналитику:

1. Требования к опыту:

– опыт работы в роли аналитика/системного аналитика на IT-проектах (особенно желателен опыт на интеграционных проектах);

– умение структурировать полученную информацию и четко излагать свои мысли;

– опыт применения методологии и нотаций моделирования предметной области (UML, EPC, BPMN);

– опыт постановки задач программистам, проектирования интеграционных решений;

– общие знания в области архитектуры web-приложений, микросервисной архитектуры, REST API;

– умение анализировать json/xsd схемы, опыт использования программ Postman/Soap UI;

– опыт разработки проектной документации в соответствии со стандартами ГОСТ 34, 19, РД-50 в сфере ИТ.

Специалист технической поддержки:

1. Требования

– техническая поддержка пользователей;

– знание ОС семейства Windows и офисных продуктов Microsoft;

– опыт администрирования рабочих станций;

– взаимодействовать с другими отделами для решения задач и проблем клиентов.

## 5 Описание жизненного цикла программного обеспечения

Описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения (ПО);

– устранение неисправностей, выявленных в ходе эксплуатации программного обеспечения;

– совершенствование программного обеспечения (ПО);

– информацию о персонале, необходимом для обеспечения такой поддержки.

### 5.1 Процессы жизненного цикла программного обеспечения

#### 5.1.1 Общие сведения

Жизненный цикл программных средств обеспечивается в соответствии с требованиями ГОСТ Р ИСО/МЭК 12207-2010. Основные процессы жизненного цикла программных средств, в соответствии с указанным ГОСТ, описаны в данном разделе.

#### 5.1.2 Процессы внедрения программных средств

##### 5.1.2.1 Основной процесс внедрения

В результате успешного осуществления основного процесса внедрения (в ГОСТ Р ИСО/МЭК 12207-2010 используется термин «реализации») программных средств:

– определяется стратегия внедрения;

– определяются ограничения по технологии реализации проекта;

– изготавливается программная составная часть;

– программная составная часть упаковывается и хранится в соответствии с соглашением о её поставке.

##### 5.1.2.2 Процесс анализа требований к программным средствам

В результате успешного осуществления процесса анализа требований к программным средствам:

– определяются требования к программным элементам системы и их интерфейсам;

– требования к программным средствам анализируются на корректность и тестируемость;

– осознается воздействие требований к программным средствам на среду функционирования;

– устанавливается совместимость и прослеживаемость между требованиями к программным средствам и требованиями к системе;

– определяются приоритеты реализации требований к программным средствам;

– требования к программным средствам принимаются и обновляются по мере необходимости;

– оцениваются изменения в требованиях к программным средствам по стоимости, графикам работ и техническим воздействиям;

– требования к программным средствам воплощаются в виде базовых линий и доводятся до сведения заинтересованных сторон.

5.1.2.3 Процессы проектирования программных средств

В результате успешной реализации процесса проектирования архитектуры программных средств:

– разрабатывается проект архитектуры программных средств и устанавливается базовая линия, описывающая программные составные части, которые будут реализовывать требования к программным средствам;

– определяются внутренние и внешние интерфейсы каждой программной составной части;

– устанавливаются согласованность и прослеживаемость между требованиями к программным средствам и программному проекту.

В результате успешного осуществления процесса детального проектирования программных средств:

– разрабатывается детальный проект каждого программного компонента, описывающий создаваемые программные модули;

– определяются внешние интерфейсы каждого программного модуля;

– устанавливается совместимость и прослеживаемость между детальным проектированием, требованиями и проектированием архитектуры.

5.1.2.4 Процесс конструирования программных средств

В результате успешного осуществления процесса конструирования программных средств:

– определяются критерии верификации для всех программных блоков относительно требований;

– изготавливаются программные блоки, определенные проектом;

– устанавливается совместимость и прослеживаемость между программными блоками, требованиями и проектом;

– завершается верификация программных блоков относительно требований и проекта.

5.1.2.5 Процесс комплексирования программных средств

В результате успешного осуществления процесса комплексирования программных средств:

– разрабатывается стратегия комплексирования для программных блоков, согласованная с программным проектом и расположенными по приоритетам требованиями к программным средствам;

– разрабатываются критерии верификации для программных составных частей, которые гарантируют соответствие с требованиями к программным средствам, связанными с этими составными частями;

– программные составные части верифицируются с использованием определенных критериев;

– программные составные части, определенные стратегией комплексирования, изготавливаются;

– регистрируются результаты комплексного тестирования;

– устанавливаются согласованность и прослеживаемость между программным проектом и программными составными частями;

– разрабатывается и применяется стратегия регрессии для повторной верификации программных составных частей при возникновении изменений в программных блоках (в том числе в соответствующих требованиях, проекте и кодах).

5.1.2.6 Процесс квалификационного тестирования программных средств

В результате успешного осуществления процесса квалификационного тестирования программных средств:

– определяются критерии для комплектованных программных средств с целью демонстрации соответствия с требованиями к программным средствам;

– комплектованные программные средства верифицируются с использованием определенных критериев;

– записываются результаты тестирования;

– разрабатывается и применяется стратегия регрессии для повторного тестирования комплектованного программного средства при проведении изменений в программных составных частях.

5.1.3 Процессы поддержки программных средств

5.1.3.1 Процесс управления документацией программных средств

В результате успешного осуществления процесса управления документацией программных средств:

– разрабатывается стратегия идентификации документации, которая реализуется в течение жизненного цикла программного продукта или услуги;

– определяются стандарты, которые применяются при разработке программной документации;

– определяется документация, которая производится процессом или проектом;

– указываются, рассматриваются и утверждаются содержание и цели всей документации;

– документация разрабатывается и делается доступной в соответствии с определенными стандартами;

– документация сопровождается в соответствии с определенными критериями.

5.1.3.2 Процесс управления конфигурацией программных средств

В результате успешного осуществления процесса управления конфигурацией программных средств:

– разрабатывается стратегия управления конфигурацией программных средств;

– составные части, порождаемые процессом или проектом, идентифицируются, определяются и вводятся в базовую линию;

– контролируются модификации и выпуски этих составных частей;

– обеспечивается доступность модификаций и выпусков для заинтересованных сторон;

– регистрируется и сообщается статус составных частей и модификаций;

– гарантируются завершенность и согласованность составных частей;

– контролируются хранение, обработка и поставка составных частей.

5.1.3.3 Процесс обеспечения гарантии качества программных средств

В результате успешного осуществления процесса гарантии качества программных средств:

– разрабатывается стратегия обеспечения гарантии качества;

– создается и поддерживается свидетельство гарантии качества;

– идентифицируются и регистрируются проблемы и (или) несоответствия с требованиями;

– верифицируется соблюдение продукцией, процессами и действиями соответствующих стандартов, процедур и требований.

5.1.3.4 Процесс верификации программных средств

В результате успешного осуществления процесса верификации программных средств:

– разрабатывается и осуществляется стратегия верификации;

– определяются критерии верификации всех необходимых программных рабочих продуктов;

– выполняются требуемые действия по верификации;

– определяются и регистрируются дефекты;

– результаты верификации становятся доступными заказчику и другим заинтересованным сторонам.

5.1.3.5 Процесс валидации программных средств

В результате успешного осуществления процесса валидации программных средств:

– разрабатывается и реализуется стратегия валидации;

– определяются критерии валидации для всей требуемой рабочей продукции;

– выполняются требуемые действия по валидации;

– идентифицируются и регистрируются проблемы;

– обеспечиваются свидетельства того, что созданные рабочие программные продукты пригодны для применения по назначению;

– результаты действий по валидации делаются доступными заказчику и другим заинтересованным сторонам.

5.1.3.6 Процесс ревизии программных средств

В результате успешного осуществления процесса ревизии программных средств:

– выполняются технические ревизии и ревизии менеджмента на основе потребностей проекта;

– оцениваются состояние и результаты действий процесса посредством ревизии деятельности;

– объявляются результаты ревизии всем участвующим сторонам;

– отслеживаются для закрытия позиции, по которым необходимо предпринимать активные действия, выявленные в результате ревизии;

– идентифицируются и регистрируются риски и проблемы.

5.1.3.7 Процесс аудита программных средств

В результате успешного осуществления процесса аудита программных средств:

– разрабатывается и осуществляется стратегия аудита;

– согласно стратегии аудита, определяется соответствие отобранных рабочих программных продуктов и (или) услуг или процессов, соответствующих требованиям, планам и соглашениям;

– аудиты проводятся соответствующими независимыми сторонами;

– проблемы, выявленные в процессе аудита, идентифицируются, доводятся до сведения ответственных за корректирующие действия и затем решаются.

5.1.3.8 Процесс решения проблем в программных средствах

В результате успешной реализации процесса решения проблем в программных средствах:

– разрабатывается стратегия менеджмента проблем;

– проблемы регистрируются, идентифицируются и классифицируются;

– проблемы анализируются и оцениваются для определения приемлемого решения (решений);

– выполняется решение проблем;

– проблемы отслеживаются вплоть до их закрытия;

– известно текущее состояние всех зафиксированных проблем.

5.2 Устранение неисправностей программного обеспечения

Перечень этапов процесса устранения неисправностей программного обеспечения (ПО) приведено в п. 4.1.3.8 «Процесс решения проблем в программных средствах».

Штатный порядок работы ПО определяется эксплуатационной документаций, предоставляемой производителем ПО. Поддерживаемый ПО набор функций определяется требованиями технического задания (ТЗ), утвержденного Заказчиком.

В случае обнаружения ошибок в работе ПО, которые являются нарушением требований ТЗ или противоречат порядку работы ПО, описанному в документации, администратор ПО должен направить заявку в службу технической поддержки (СТП) организации, проводившей работы по внедрению ПО. СТП организации, внедрившей ПО, проверяет, при необходимости уточняет полученную заявку и пытается выполнить ее, использую собственные ресурсы и знания.

В случае, если силами СТП организации, внедрившей ПО, выполнить заявку не удается, то указанная организация обращается за помощью к производителю ПО. Производитель ПО проверяет наличие ошибки и рекомендаций по ее устранению в базе знаний технической поддержки (в соответствии с договорённостями между производителем и заказчиком).

После устранения неисправности разработчики ПО выпускают обновление к текущей версии ПО или включают исправление в следующую версию ПО. Информация о наличии обновления или новой версии ПО доводится до партнёров производителя ПО. В случае наличия у Заказчика контракта или договора на поддержку ПО, Заказчик имеет право на получение обновления ПО.

## 6 Совершенствование программного обеспечения

Работа по совершенствованию ПО включает в себя два основных направления:

- повышение качества и надежности ПО;
- актуализация перечня функций, поддерживаемых ПО.

В ходе постоянно проводимой работы по совершенствованию ПО используются хорошо зарекомендовавшие себя методы повышения качества и надежности ПО:

- совершенствование процесса разработки ПО – повышение качества ПО за счёт использования современных методик и инструментов разработки;
- совершенствование процесса тестирования ПО – обеспечение необходимой полноты покрытия.

Актуализация перечня функций, поддерживаемых ПО, включает в себя:

- добавление новых и изменение существующих функций в соответствии со стратегией развития ПО;
- добавление новых и изменение существующих функций по предложениям Заказчиков и партнёров производителя ПО;
- исключение устаревших функций.

## 7 Документирование

Документирование продукта обеспечивается:

– документирование программного обеспечения, подготовленного разработчиками (API).

– подготовкой эксплуатационной документации.