Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра информационных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №1

по дисциплине «Коллективная разработка приложений»

Выполнили: ст. гр. 4ИТ

Хижний Е.Г., Большаков М.О., Сторчак В.В.

Проверил: доц. каф. ИТ

Полетайкин А.Н.

Краснодар

2022

## Лабораторная работа №1

**Тема: Создание командного проекта**

**Цель лабораторной работы**: получить практические навыки в создании инфраструктуры командного проекта, а также в создании и администрировании команды проекта.

**Тема проекта:** Электронная зачетная книжка на основе REST API Service.

### Задание

1. Выполнить подключение к Azure DevOps Server через персональную учетную запись Microsoft. При необходимости создать её.
2. При помощи панели «Командный обозреватель» создать командный проект по разработке программного приложения и выполнить его первичную настройку

**Ход работы:**

Подключимся к Azure DevOps Server. Для этого запустим интегрированную среду Visual Studio 2019. Вызовем Team Explorer во вкладке Вид.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Вызов Team Explorer

Далее подключимся к Azure DevOps. Для этого кликнем по кнопке “Подкючиться…”.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Team Explorer

Продолжим работу с рабочим аккаунтом Microsoft, в который выполнен вход.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Вход в аккаунт

Укажем наименование команды, местоположение и введем проверочный код.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Создание команды

Далее создадим проект “Журнал оценок”, который будет доступен только тем, у кого есть ссылка.

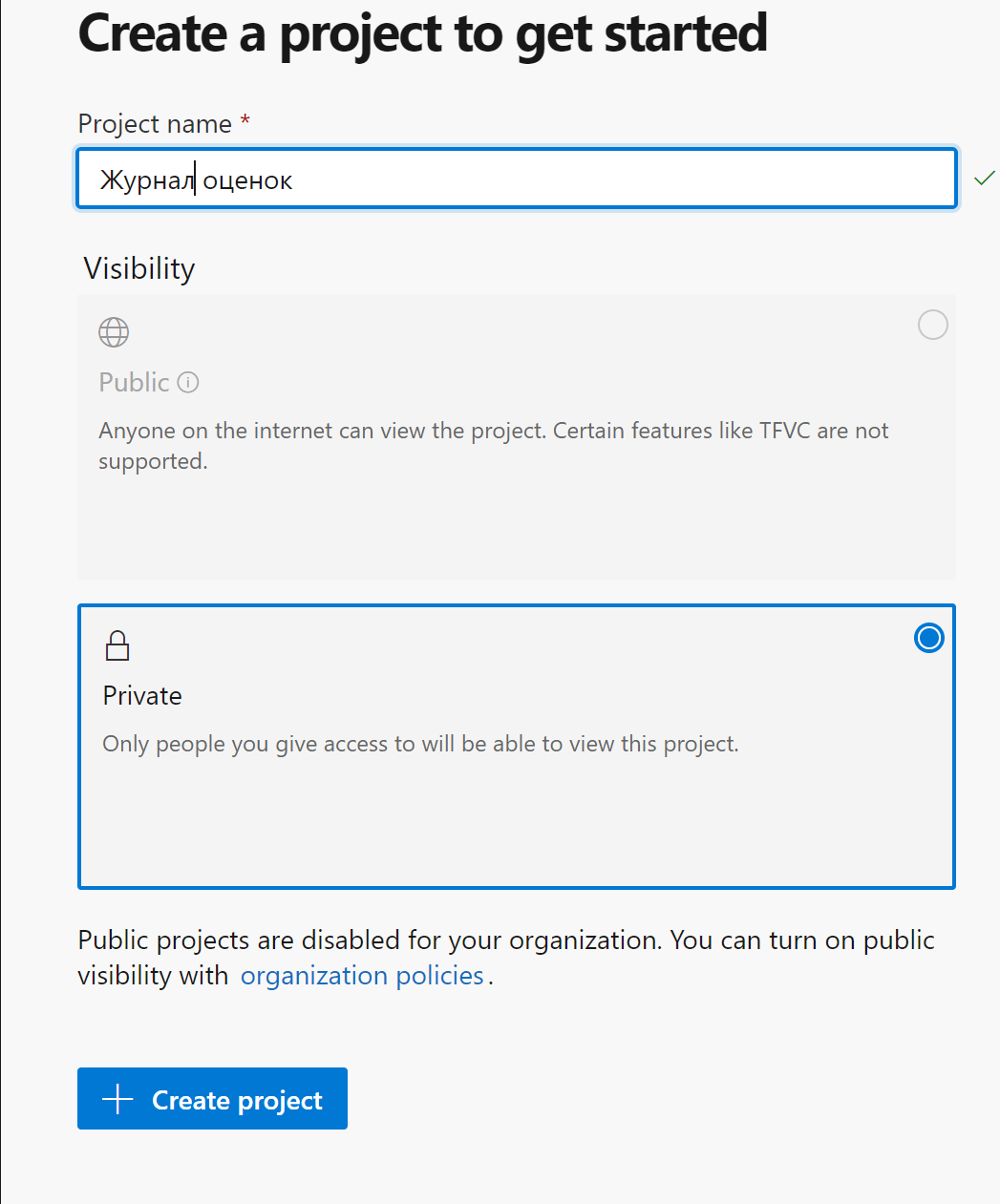


Рисунок 5 – Создание проекта

Страница проекта выглядит следующим образом.

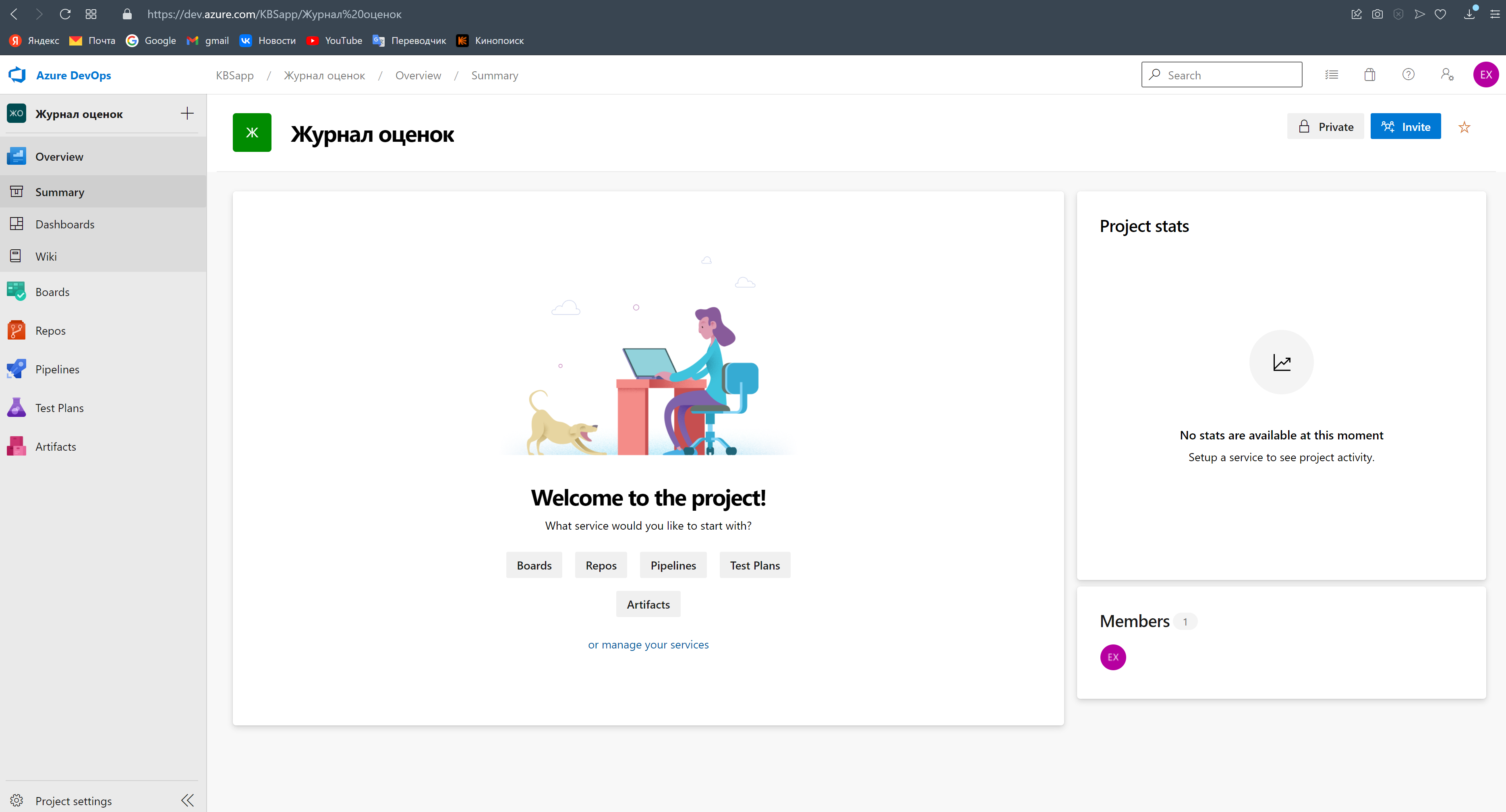


Рисунок 6 – Страница проекта

Подключимся к проекту в Visual Studio. При нажатии на кнопку “Обновить” появится созданная команда и проект. Выберем наш проект и нажмем “Клонировать”.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Подключение к проекту

В параметрах Visual Studio сменим систему управления версиями с Git на Visual Studio Team Foundation Server.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Изменение системы управления версиями

**Вывод:** в ходе работы были получены практические навыки в создании инфраструктуры командного проекта, а также в создании и администрировании команды проекта.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра информационных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №2

по дисциплине «Коллективная разработка приложений»

Выполнили: ст. гр. 4ИТ

Хижний Е.Г., Большаков М.О., Сторчак В.В.

Проверил: доц. каф. ИТ

Полетайкин А.Н.

Краснодар

2022

## Лабораторная работа №2

**Тема: Настройка параметров командного проекта**

**Цель лабораторной работы:** получить практические навыки в создании и администрировании команды проекта

**Тема проекта:** Электронная зачетная книжка на основе REST API Service.

### Задание

1. Спланировать Backlog проекта, используя методологию Scrum.
2. Выполнить настройку области и итераций командного проекта. Определить 3 спринта, длительностью 24, 18 и 12 дней соответственно.
3. Создать и настроить команду проекта, состоящую не менее чем из трех человек.
4. Выполнить настройку оповещений командного проекта.
5. Протестировать веб-интерфейс Team Web Access командного проекта и убедиться в его работоспособности.

**Ход работы:**

Перейдем в веб-интерфейс Team Web Access командного проекта. На вкладке Sprints создадим 3 спринта длительностью в 24, 18, 12 дней.

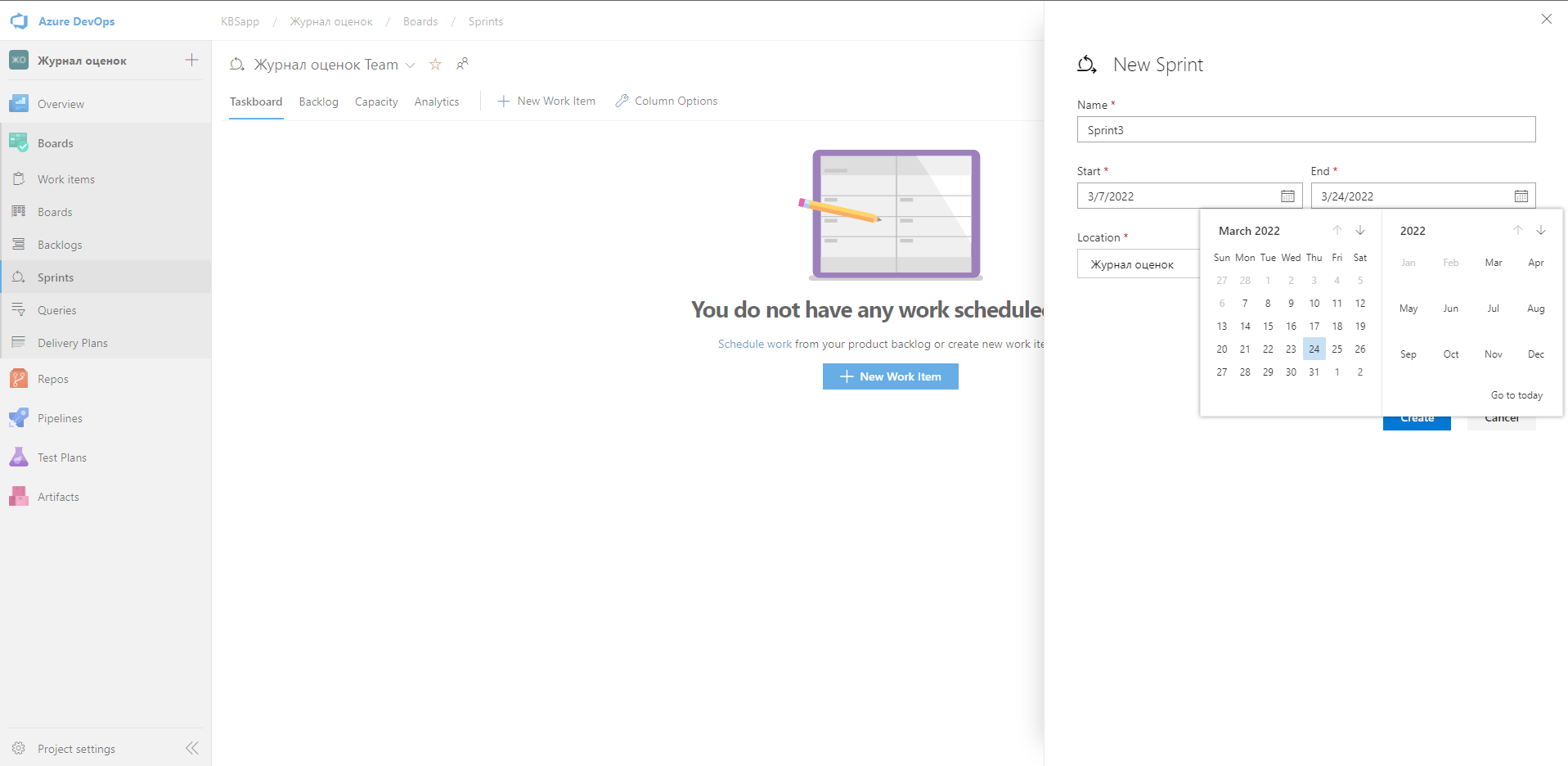


Рисунок 1 – Вкладка Sprints

В итоге получим такой журнал спринтов.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Журнал спринтов

Также выставим рабочие дни для спринтов в настройках

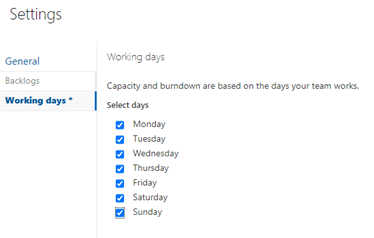


Рисунок 3 – Настройка спринтов

Теперь добавим всех участников в организацию. Перейдем в настройки организации, во вкладку Users. Кликнем по кнопке Add Users.

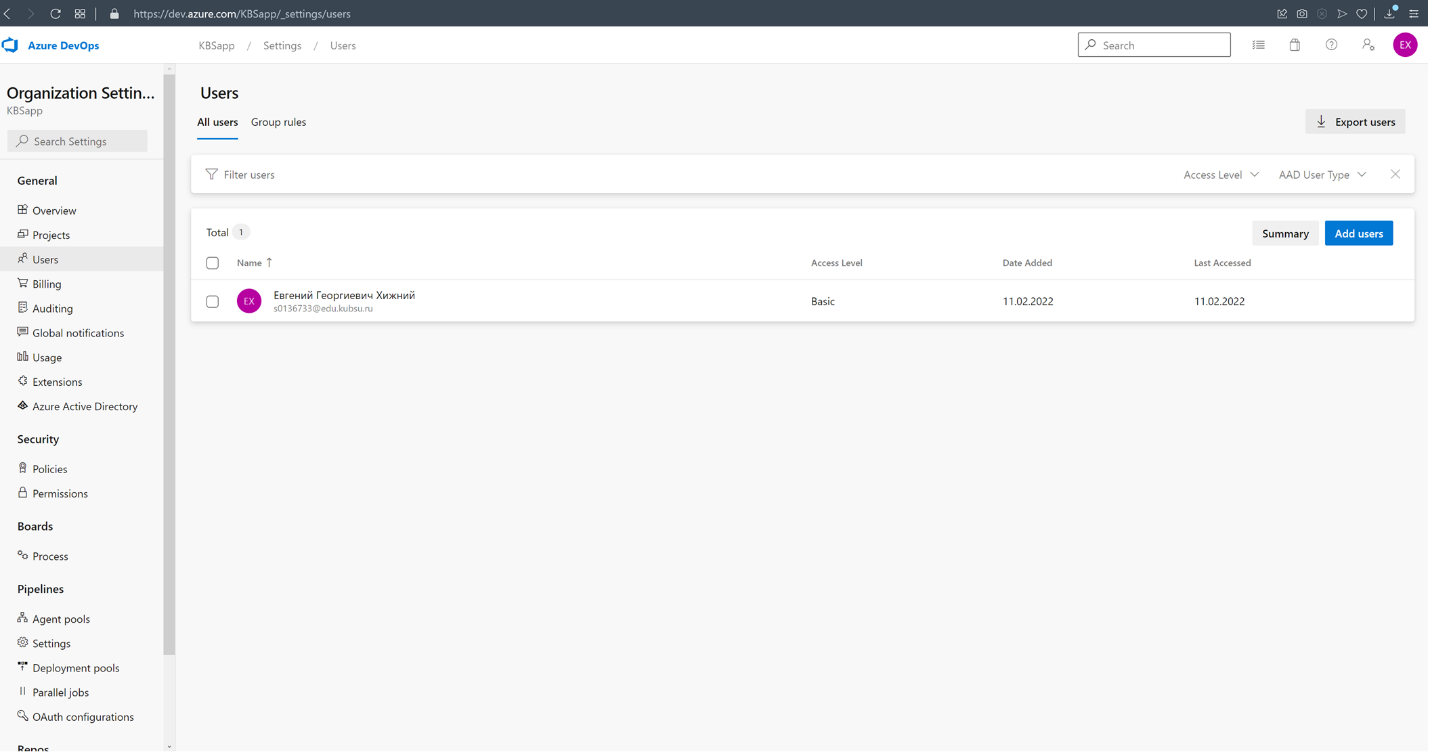


Рисунок 4 – Настройки организации

Введем электронную почту участников, укажем проект и группу.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Добавление пользователей

На почту по указанным адресам отправлено сообщение с приглашением на присоединение к организации.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Приглашение в организацию

В результате окно пользователей выглядит следующим образом.

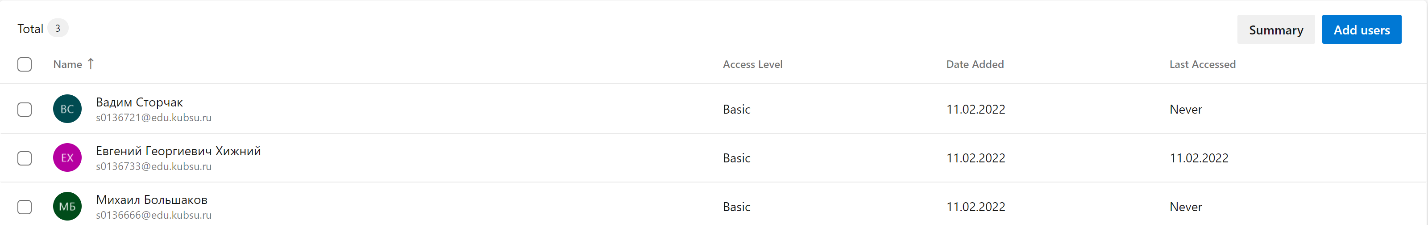


Рисунок 7 – Добавленные пользователи

Перейдем на вкладку настройки оповещений и выберем необходимые опции.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Настройки оповещений

**Вывод:** в ходе работы были получены практические навыки в создании и администрировании команды проекта.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра информационных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №3

по дисциплине «Коллективная разработка приложений»

Выполнили: ст. гр. 4ИТ

Хижний Е.Г., Большаков М.О., Сторчак В.В.

Проверил: доц. каф. ИТ

Полетайкин А.Н.

Краснодар

2022

## Лабораторная работа №3

**Тема: Разработка требований к программному приложению**

**Цель лабораторной работы:** получить практические навыки в разработке пользовательских требований к программному продукту с помощью Visual Studio, Team Web Access, Microsoft Excel и Microsoft Project.

**Тема проекта:** Электронная зачетная книжка на основе REST API Service.

### Задание

1. Изучить теоретический материал.
2. Для разрабатываемого проекта создать пользовательские требования с использованием следующих инструментальных средств:

* Team Web Access (не менее двух требований);
* Microsoft Visual Studio (не менее двух требований);
* Microsoft Excel(заменен на Trello) (не менее одного требования).

**Ход работы:**

### Team Web Access

На рис.1 показан пример создания задачи в Team Web Access. Ее реквизиты:

* Наименование – Построить модель Базы данных;
* Ответственное – Вадим Сторчак;
* Приоритет – 3;
* Роль (Activity) – Development;
* Примечание – Разработать модель Базы данных для хранения Журналов Посещаемости и Оценок студентов по предметам.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.2 – Создание задачи в Team Web Access

Просмотр задач показан на рис.2.2.

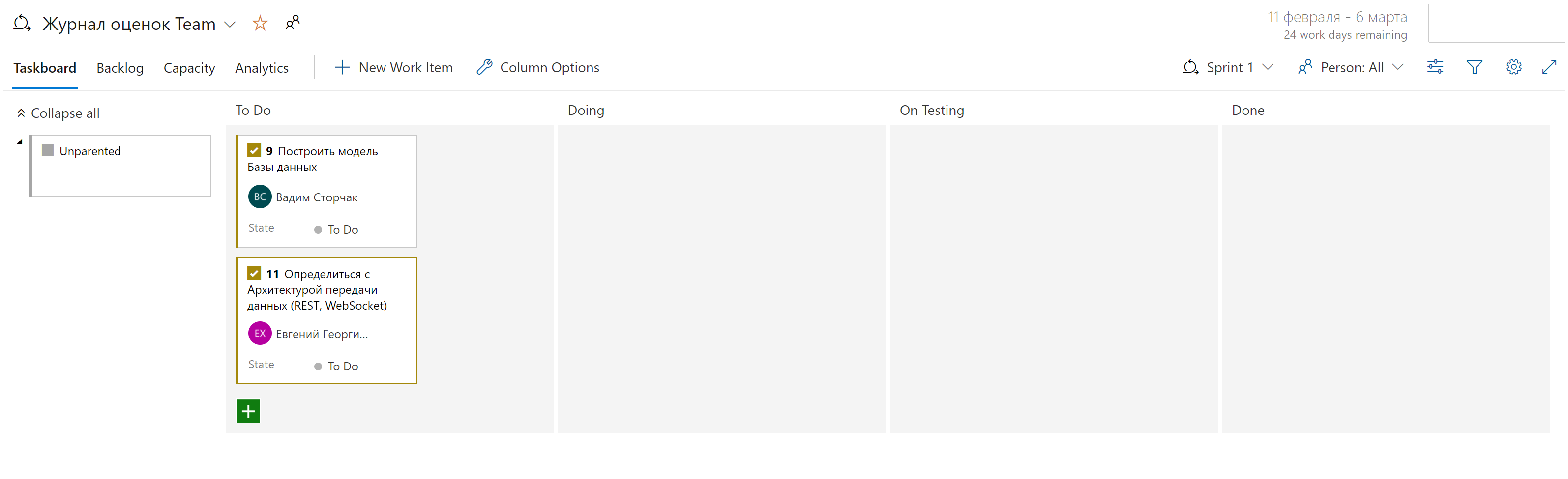


Рисунок 1.2 – Доска состояний задач

### Microsoft Visual Studio

Для создания задачи средствами Microsoft Visual Studio необходимо в верхнем меню выбрать Команды>Создать рабочий элемент>Task, как показано на рис.2.1.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.1 – Открытие окна создания задачи в Microsoft Visual Studio

На рис.2.2 показан пример создания задачи в Microsoft Visual Studio, он все реквизиты и методы работы аналогичны Team Web Access.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.2 – Создание задачи в Microsoft Visual Studio

### Trello

Trello — облачная программа для управления проектами небольших групп.

Создадим новое рабочее пространство, как показано на рис. 3.1

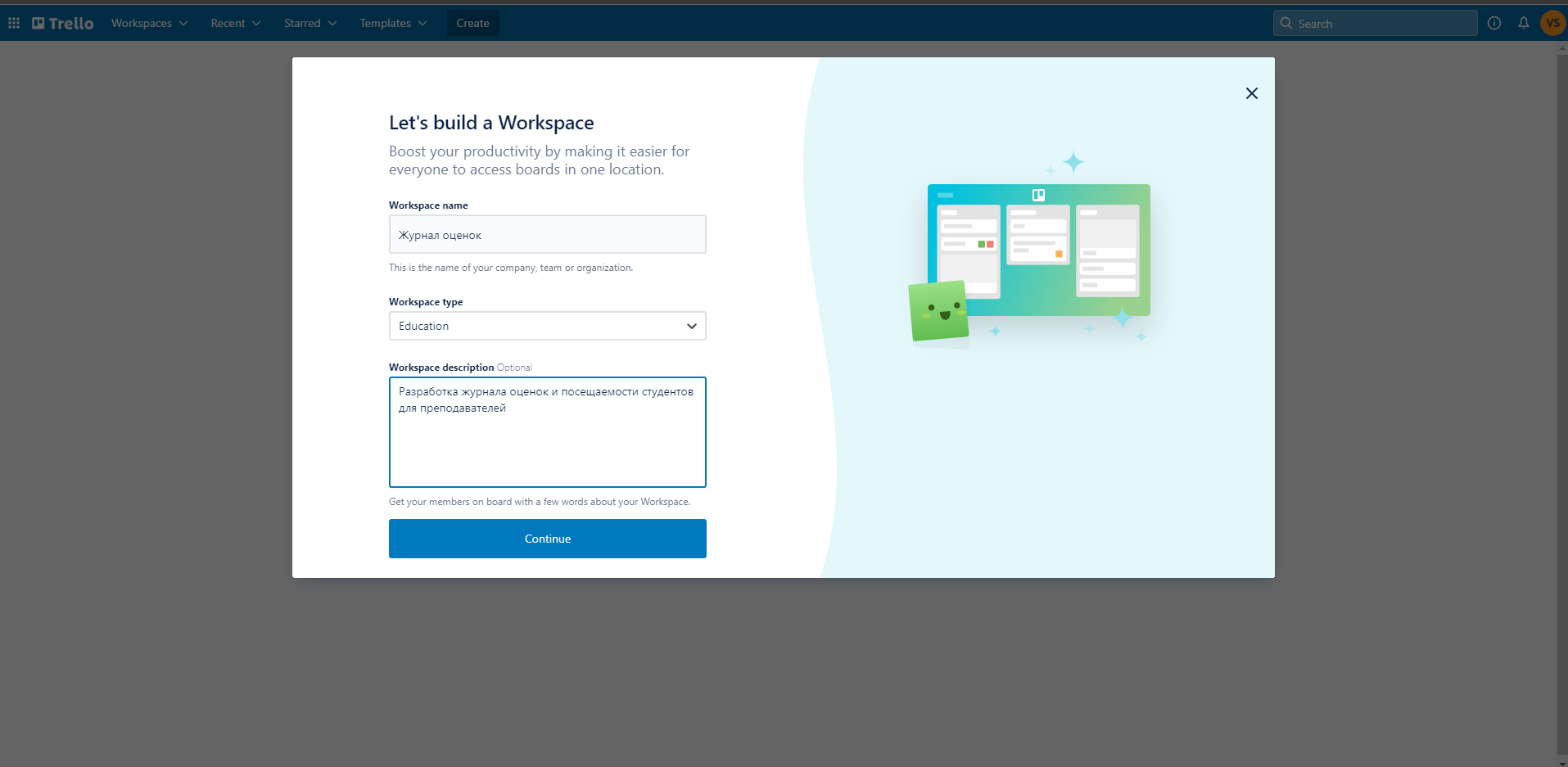


Рисунок 3.1 – Окно создания рабочего пространства в Trello

На рис.3.2 показано созданное пустое рабочее пространство.

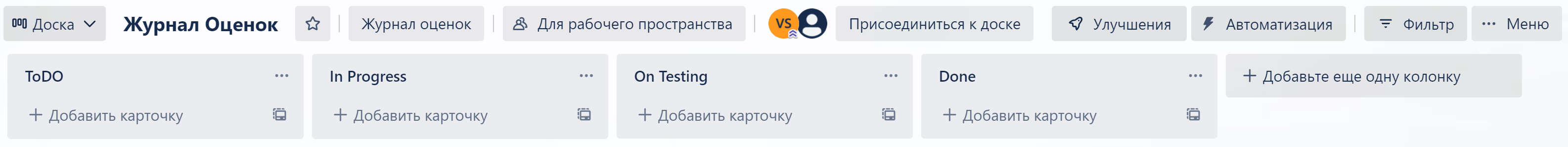


Рисунок 3.2 – Пустое рабочего пространства в Trello

Для создания новой задачи в выбранном списке нажимаем на «…»>Добавить карточку…, как показано на рис. 3.3.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.3 – Меню списка задач

На рис.3.4. показано окно создания задачи.

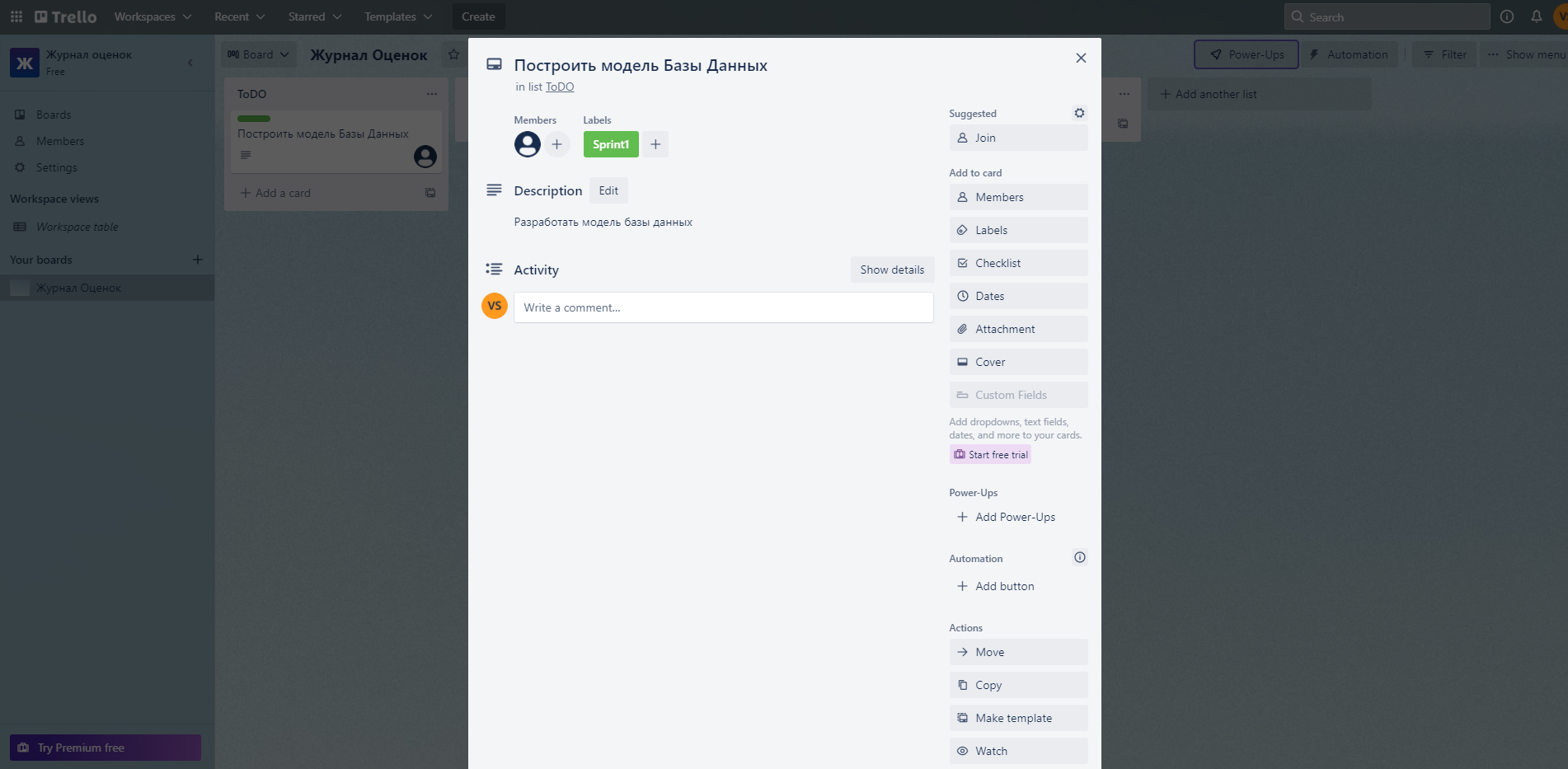


Рисунок 3.4 – Создание задачи в Trello

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.5 – Рабочее пространства в Trello с задачами

**Вывод:** в ходе лабораторной работы были получены практические навыки в разработке пользовательских требований к программному продукту с помощью Visual Studio, Team Web Access, Trello.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра информационных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №4

по дисциплине «Коллективная разработка приложений»

Выполнили: ст. гр. 4ИТ

Хижний Е.Г., Большаков М.О., Сторчак В.В.

Проверил: доц. каф. ИТ

Полетайкин А.Н.

Краснодар

2022

## Лабораторная работа №4

**Тема: Планирование итераций**

**Цель лабораторной работы:** получить практические навыки планирования итераций.

**Тема проекта:** Электронная зачетная книжка на основе REST API Service.

### Задание

1. Провести оценку сложности элементов работы и внести оценки трудозатрат для каждой задачи проекта.
2. Установить всем рабочим элементам адекватные приоритеты.
3. Для рабочих элементов, включенных в текущий спринт, детализировать задачи.
4. Провести оценку трудозатрат для задач спринта.
5. Распределить задачи спринта между членами команды.
6. Проанализировать загруженность членов команды. При необходимости выполнить выравнивание нагрузки.

**Ход работы:**

Перейдем на вкладку доски задач проекта.

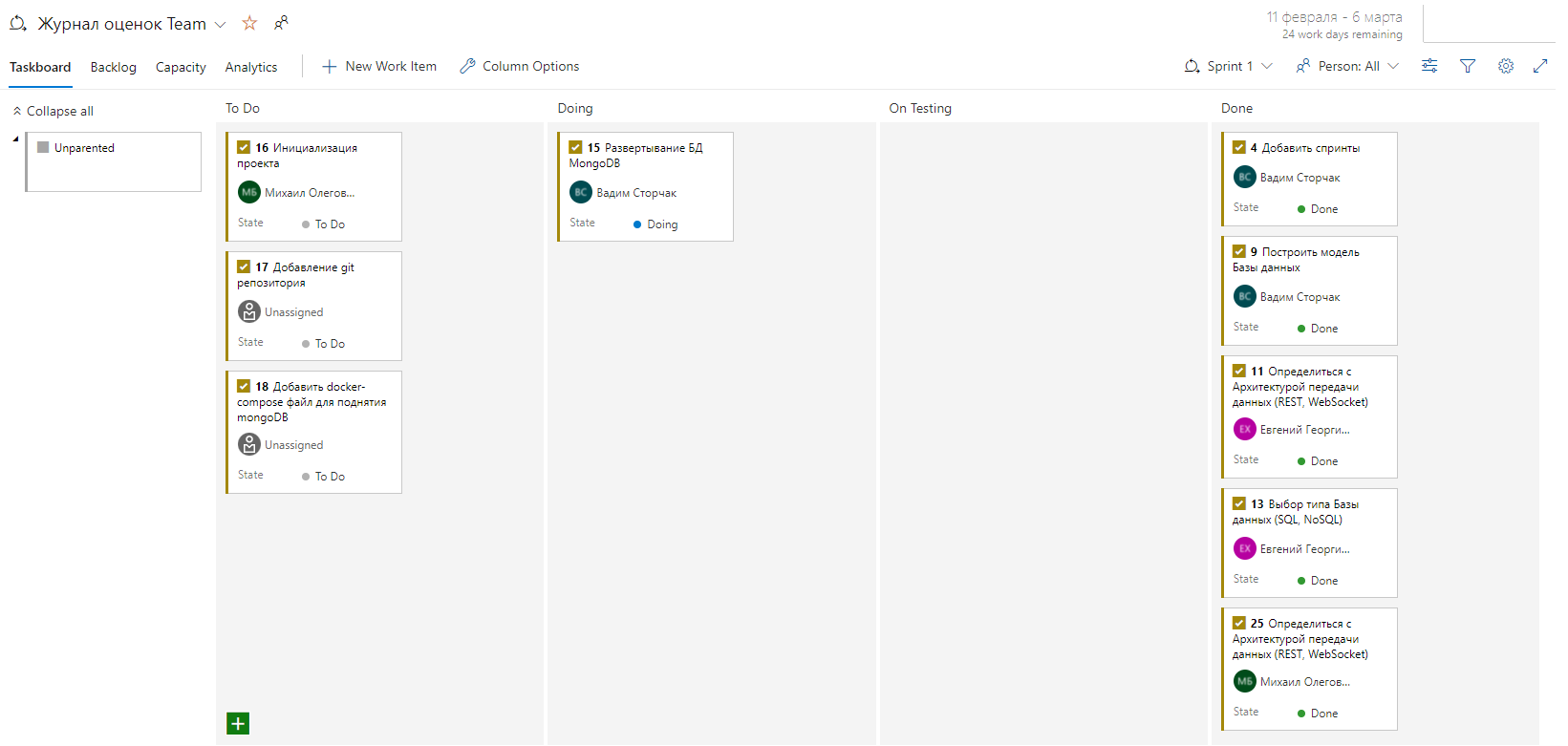


Рисунок 1 – Доска задач

Добавим для каждой задачи приоритет и объем работы. Для этого укажем данные в полях Priority и Remaining Work. Также подкорректируем состояния задач.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Редактирование задачи

В итоге получим следующую структуру для каждого спринта.



Рисунок 3 – Спринт 1

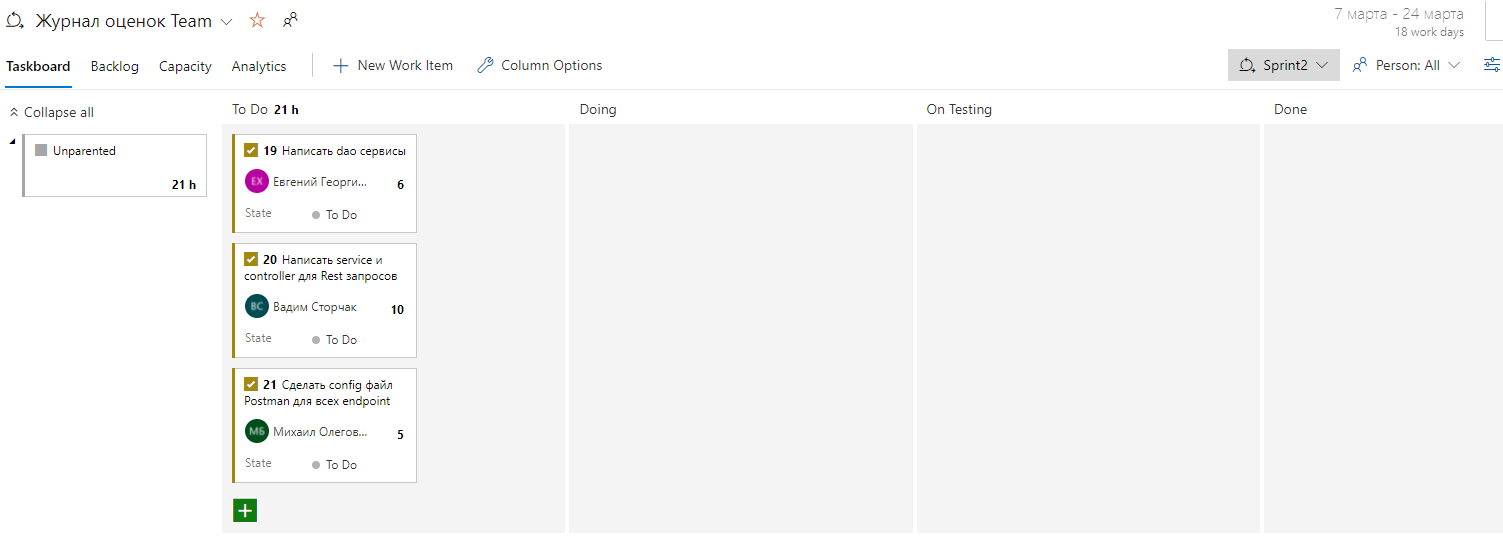


Рисунок 4 – Спринт 2

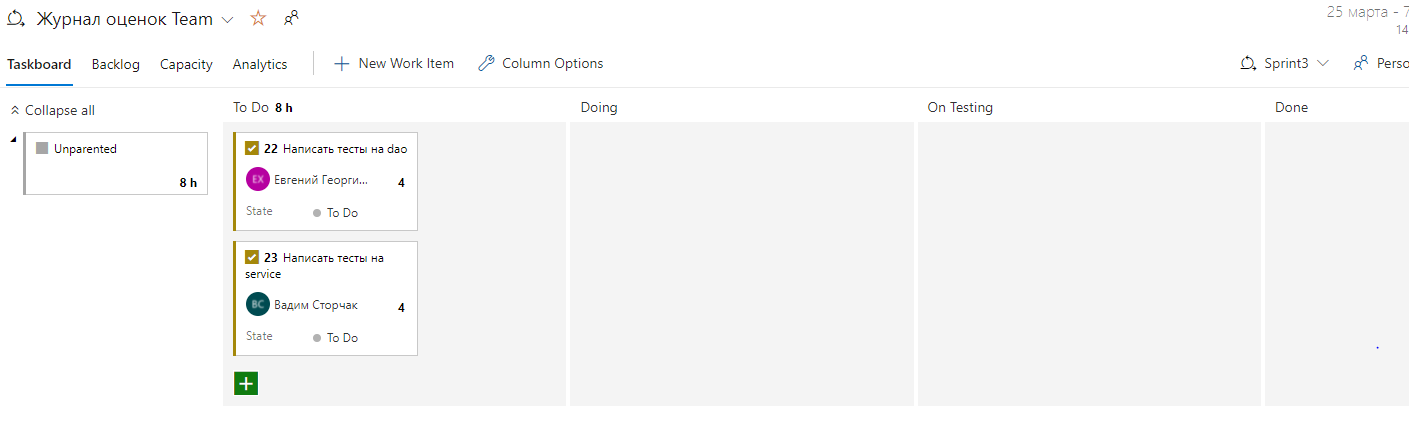


Рисунок 5 – Спринт 3

Для некоторых задач также введем подзадачи. Например, для задачи Инициализации проекта введем 3 дочерних элемента, один существующий – Выбор типа Базы данных и 2 новых.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Создание дочерних элементов

Далее составим трудоспособность членов команды, зададим для каждого пользователя деятельность и трудозатраты.



Рисунок 7 – Трудозатраты членов команды

Далее просмотрим результаты оценки затрат на реализацию задач спринта 1.

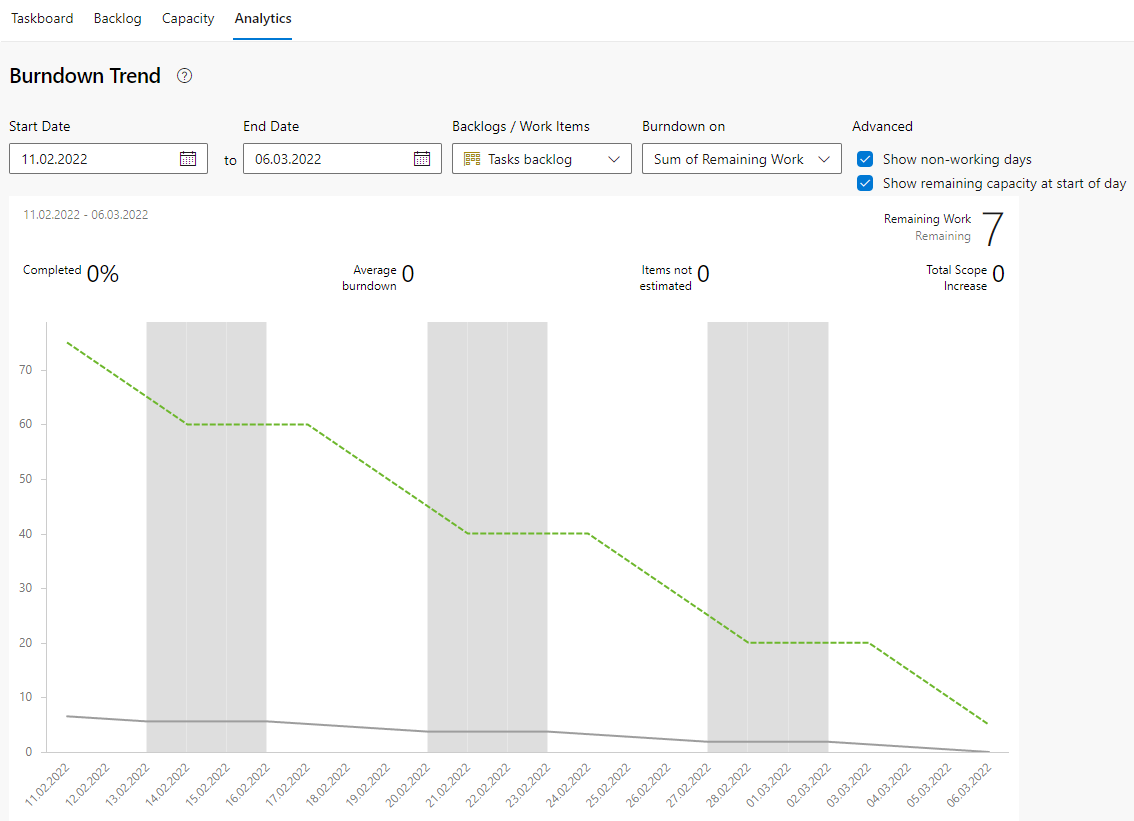


Рисунок 8 – График для оставшейся работы

Для просмотра задач конкретного пользователя перейдем на вкладку Queries. Выберем My Queries и Assigned to me. Здесь можно просмотреть задачи, реализуемые определенным членом команды.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Просмотр задач пользователя

**Вывод:** в ходе работы были получены практические навыки планирования итераций.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра информационных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №5

по дисциплине «Коллективная разработка приложений»

Выполнили: ст. гр. 4ИТ

Хижний Е.Г., Большаков М.О., Сторчак В.В.

Проверил: доц. каф. ИТ

Полетайкин А.Н.

Краснодар

2022

### Лабораторнаяработа №5

**Тема: Работа с базой данных в автономном режиме**

**Цель лабораторной работы**: получить практические навыки разработки, тестирования, построения и развертывания базы данных приложения в автономном режиме.

**Тема проекта:** Электронная зачетная книжка на основе REST API Service.

### Задание

1. При помощи инструментального средства SQL Server Management Studio создать базу данных, спроектированную при изучении дисциплины «ТППО».
2. Создать в MS Visual Studio проект базы данных и импортировать в него объекты БД, разработанной при выполнении п.1.
3. Выполнить возврат изменений на сервер.
4. По согласованию с преподавателем провести модификацию базы данных.
5. Сгенерировать скрипт для создания тестовой базы данных.
6. При помощи утилиты SQL Server Management Studio создать тестовую базу данных посредством выполнения сгенерированного скрипта.
7. Сформировать диаграмму тестовой базы данных.

**Ход работы**

В данной лабораторной работе нами была выбрана база типа NoSQL (– это подход к реализации масштабируемого хранилища (базы) информации с гибкой моделью данных, отличающийся от классических реляционных СУБД) – MongoDB (документоориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных).

В данной работе были разработаны следующие коллекции:

1. Subject (Предметы) – Содержит:
   1. Id;
   2. Name – название;
   3. Description – описание;
2. Student (Студенты) – Содержит:
   1. Id;
   2. first\_name – имя;
   3. last\_name – фамилия;
   4. middle\_name – отчество;
   5. course – курс;
   6. group\_id - id группы;
   7. phone – телефон;
3. Group (Группа) – Содержит:
   1. List\_id\_subject – список id предметов;
   2. Name – название;
   3. Faculty – факультет;
4. Teachers (Преподаватели) – Содержит:
   1. Id;
   2. first\_name – имя;
   3. last\_name – фамилия;
   4. middle\_name – отчество;
   5. phone – телефон;
   6. address – адрес;
   7. job – должность;
   8. science\_degree – учёная степень;
   9. List\_id\_groups – список id групп;
   10. List\_id\_subject – список id предметов;
5. Mark (Оценки по экзаменам) – Содержит:
   1. Id\_sbject - id предмета, по которому поставили оценку;
   2. Id\_student – id студента, которому поставили оценку;
   3. Id\_teacher – id преподавателя, который поставил оценку;
   4. Mark – оценка.

Структура коллекций показана на рис.1

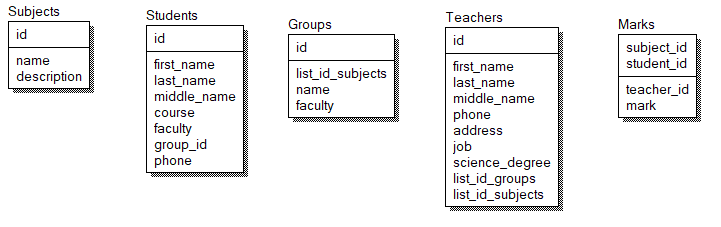


Рисунок 1 – Структура коллекций БД

База данных была развернута на удаленном сервере при помощи облачной платформы Amazon web server (<https://aws.amazon.com/ru/>). На данном сервисе была поднята виртуальной машина с операционной системой Ubuntu server 20.04 LTS. На рис.2 показаны параметры ВМ.

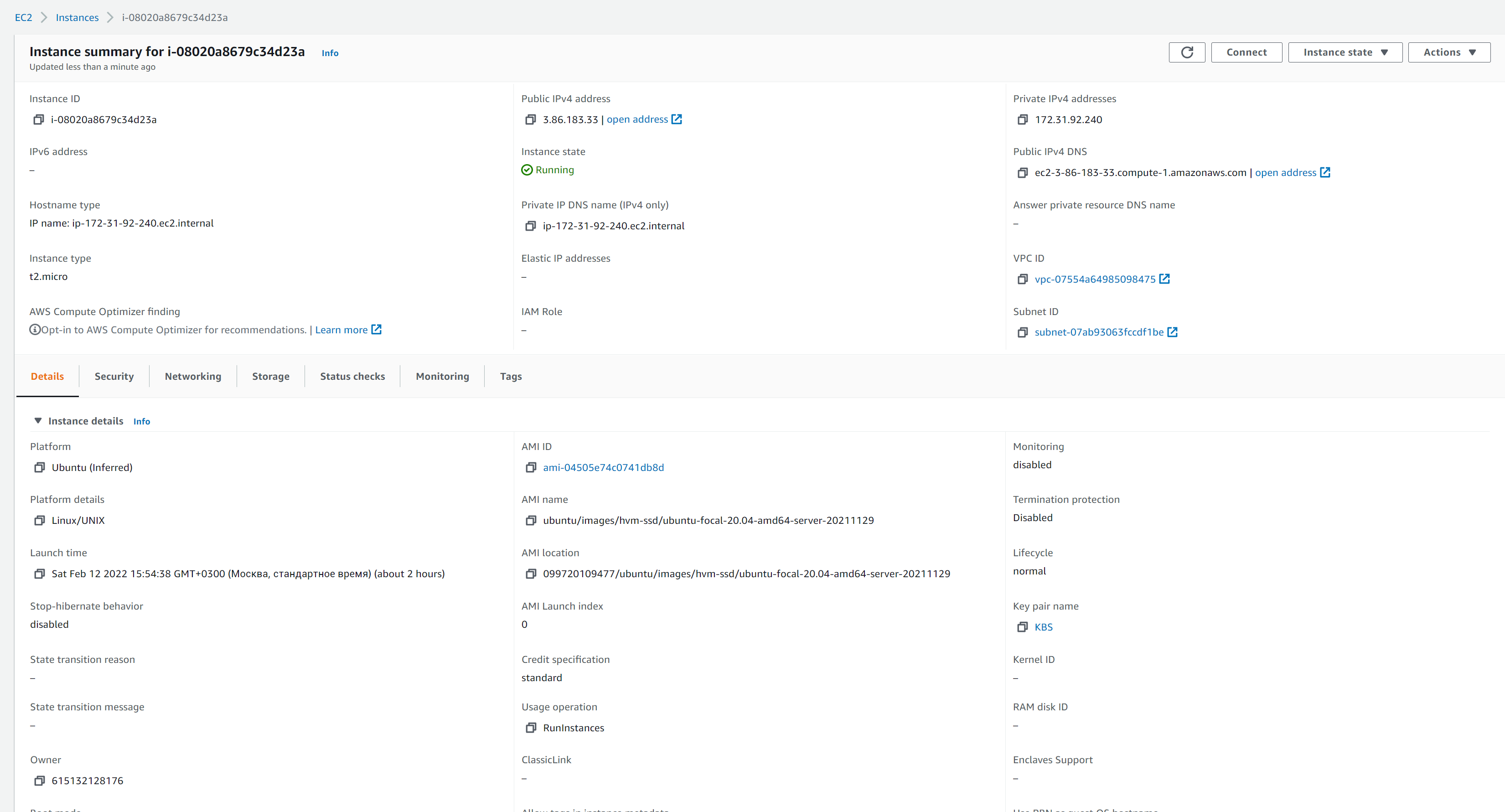


Рисунок 2 – Параметры виртуальной машины

Подключение к виртуальной машине происходит по SSH. На рис.3 показано подключение к ВМ с локального компьютера.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Подключение к виртуальной машине

Для поднятия MongoDB на удалённом сервере использовалось средство контейнеризации Docker. Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

Для работы с базой данных на сервере достаточно прокинуть тоннель до сервера с портом 27017.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Код docker-compose.yaml

**Вывод:** в ходе лабораторной работы были получены практические навыки разработки, тестирования, построения и развертывания базы данных приложения в автономном режиме.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра информационных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №6

по дисциплине «Коллективная разработка приложений»

Выполнили: ст. гр. 4ИТ

Хижний Е.Г., Большаков М.О., Сторчак В.В.

Проверил: доц. каф. ИТ

Полетайкин А.Н.

Краснодар

2022

## Лабораторная работа №6

**Тема: Разработка и анализ приложения**

**Цель лабораторной работы:** получить практические навыки в управлении этапом разработки кода программных компонентов.

**Тема проекта:** Электронная зачетная книжка на основе REST API Service.

### Задание

1. Перевести задачи командного проекта в состояние **In Progress**.
2. Создать в решении проект для разработки приложения ИС.
3. Провести необходимые работы по этапу разработки кода приложения ИС в составе командного проекта.
4. Провести анализ кода приложения.

**Ход работы**

В ходе обсуждения были распределены задачи для каждого участника команды. На рис. 1–3 показаны задачи каждого участника команды.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Задачи Вадима Сторчака

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Задачи Михаила Большакова

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Задачи Евгения Хижнего

Проект был создан в среде разработки IntelliJ Idea c подключение git-репозитория из AzureDevOps. Окно проекта в IntelliJ Idea показан на рис. 4.

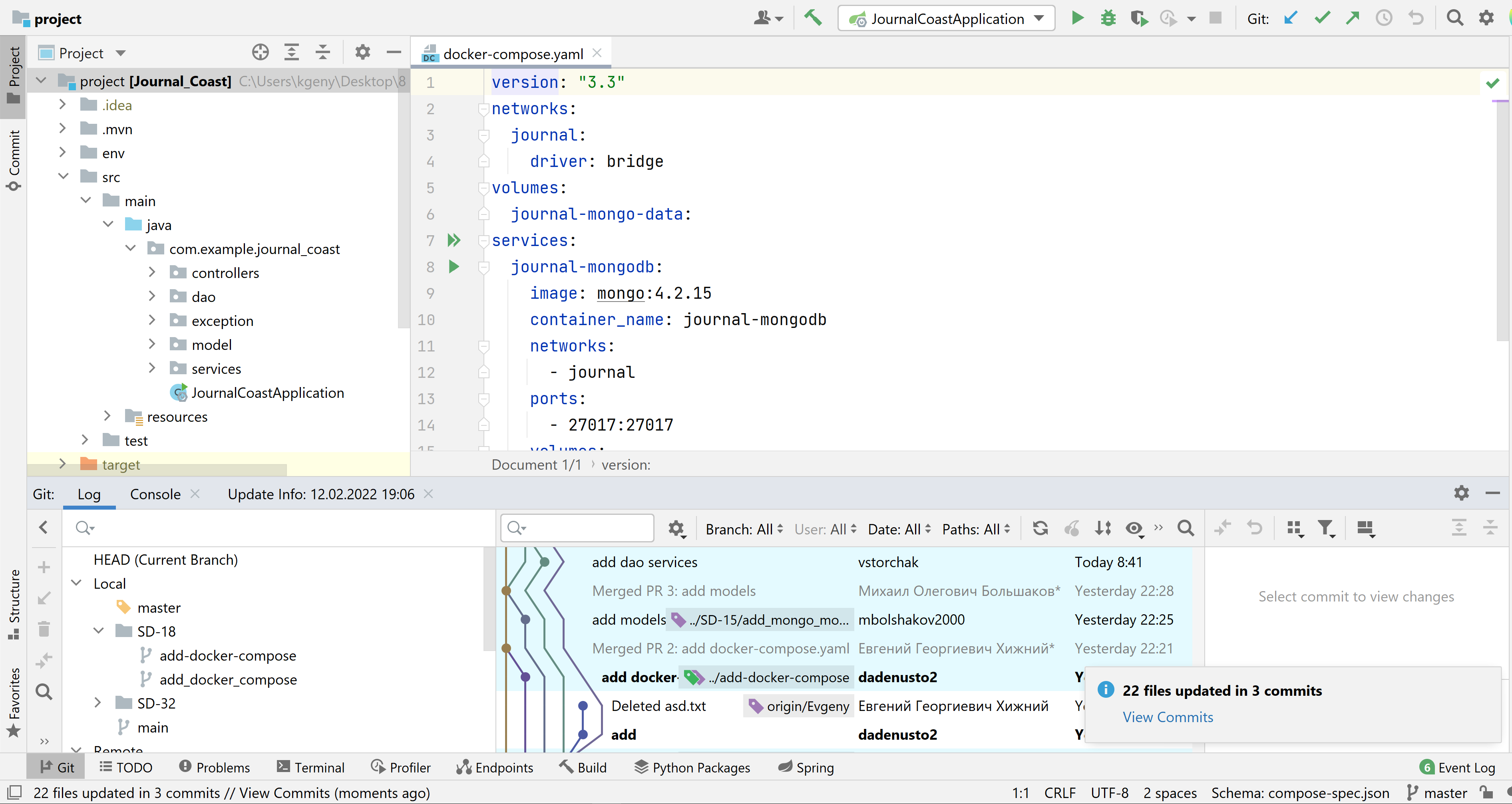


Рисунок 4 – Окно проекта в IntelliJ Idea

Основной веткой проекта была ветка master. Для решения каждой задачи в проекте, связанной с написанием кода, участником команды создавалась соответствующая ветка от master, выполнялась задача и отправлялась на проверку другому участнику команды для вливания кода (merge) в основную ветку.

Для вливания задачи создадим ветку для этой задачи, внесем в нее изменения согласно задаче и отправим изменения на удалённый git-репозиторий. В Team Web Access перейдем в Repos>Pull request. Появиться наша ветка. Для создания запроса на вливания нажмем на “New pull request” (Рис. 5).

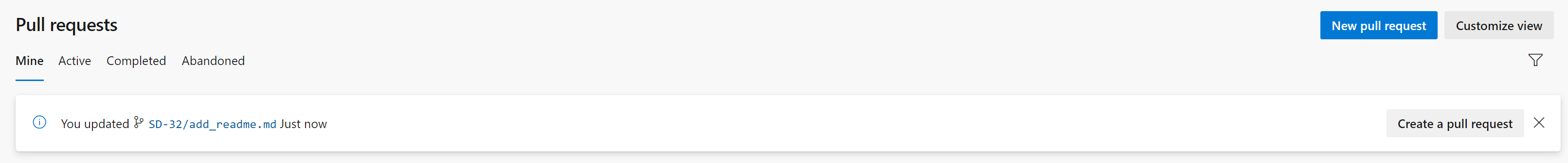


Рисунок 5 – Задачи на вливание

В открывшемся окне (рис. 6) вводим:

1. выбираем какую ветку вливаем;
2. выбираем ветку, в которую вливаем;
3. название;
4. описание;
5. участника команды, который проверит задачу;
6. задачу из списка зада, к которой относится данный запрос.

Для создания запроса на вливания нажмем на “Create”.

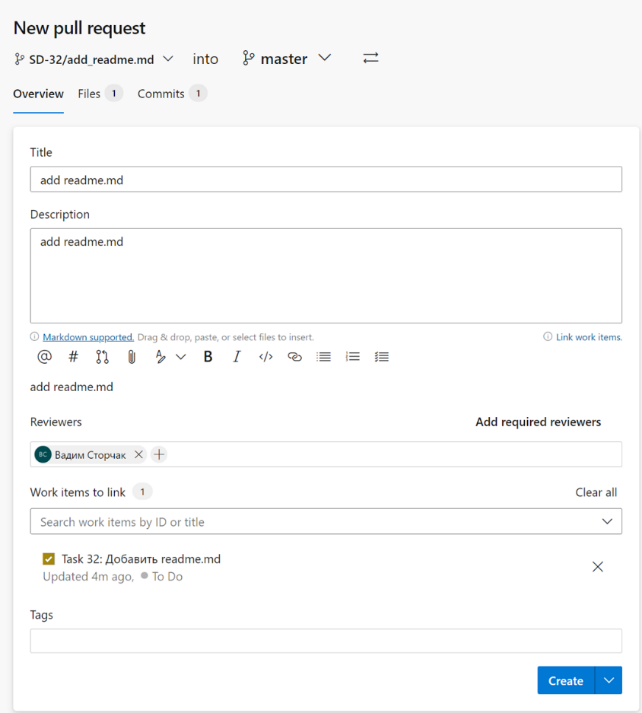


Рисунок 6 – Окно создания запроса

Далее необходимо дождаться проверки задачи и добавить ее в ветку (Кнопка “Complete”) (Рис.7). Сам запрос помечен как “Active”.

Изображение выглядит как текст, снимок экрана, внутренний, ноутбук

Автоматически созданное описание

Рисунок 7 –Запрос на вливание

На рис.8 показано окно для вливания задачи. Можем удалить данную ветку после завершения, добавить название вливания и его описании.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Вливание задачи

На рис. 9 показано, что вливание прошло успешно

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 –Успешное вливание

Каждый участник команды проделывает данные действия с задачами, связанными с написанием кода.

На рис.10-12 приведен анализ кода.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Анализ зависимостей в проекте и по классам

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – Анализ кода на Warning

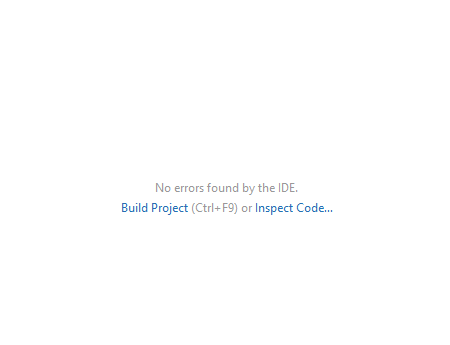


Рисунок 12 – Проверка на ошибки в проекте

**Вывод:** В данной лабораторной работы были получены практические навыки в управлении этапом разработки кода программных компонентов.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра информационных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №7

по дисциплине «Коллективная разработка приложений»

Выполнили: ст. гр. 4ИТ

Хижний Е.Г., Большаков М.О., Сторчак В.В.

Проверил: доц. каф. ИТ

Полетайкин А.Н.

Краснодар

2022

## Лабораторная работа №7

**Тема: Автоматическое тестирование**

**Цель лабораторной работы:** получить практические навыки создания автоматических тестов и привязки их к тестовым случаям.

### Задание

1. Создать автоматический тест для проверки требования, обладающего наиболее выраженной функциональностью.
2. Привязать автоматический тест к тестовому случаю проекта.

**Ход работы**

**1 Модульное тестирование**

Так как backend написан на Java Spring, для тестирования подключим основные библиотеки – Mockito, Junit и Hamcrest. Для того, чтобы не поднимать весь контекст приложения, а тестировать только отдельные модули, будем использовать Mockito. При помощи аннотации @Mock создаётся объект «заглушка», который не будет иметь никакого функционала, но будет иметь интерфейс перехвата методов. Рассмотрим на примере сервиса по получению Оценок – MarkServiceTest, листинг которого приведён ниже:

@ExtendWith(MockitoExtension.class)  
public class MarkServiceTest {  
 @Mock  
 private MarkDao markDao;  
  
 @InjectMocks  
 private MarkService markService;  
  
 @Before  
 public void before() {  
 MockitoAnnotations.*initMocks*(this);  
 }  
  
 private final Mark mark = Mark.*builder*()  
 .id("id")  
 .mark(5)  
 .build();  
  
 @Test  
 public void getMark() {  
 *when*(markDao.getMark(*eq*(mark.getId()))).thenReturn(mark);  
  
 Mark result = markService.getMark(mark.getId());  
  
 *verify*(markDao, *times*(1)).getMark(*eq*(mark.getId()));  
 *assertThat*(result, *equalTo*(mark));  
 }  
  
 @Test  
 public void saveMark() {  
  
 markService.saveMark(mark);  
  
 *verify*(markDao, *times*(1)).saveMark(*eq*(mark));  
 }  
  
 @Test  
 public void deleteMark() {  
  
 markService.deleteMark(mark.getId());  
  
 *verify*(markDao, *times*(1)).deleteMark(*eq*(mark.getId()));  
 }  
}

Сервис очень простой, он имеет один inject объект markDao, через который должен производиться запрос в базу данных. Все тесты должны быть написаны по семантическому правилу:

Тест public nameTest(){

1. Подготовка данных (when, создание объектов)
2. Выполнили тестируемый метод
3. Проверим результат

}

На основе этого правила написан тест MarkServiceTest.

verify(SomeMock, times(CountOfCall)).callingMethod(eq(param)); проверяет, что в рамках проверяемого метода должна вызваться callingMethod в кол-ве CountOfCall раз.

Если тестируемый метод возвращает какой-либо объект, то можно проверить его на соответствие поставки логики работы метода, для это воспользуемся методами библиотек Junit и Hamcrest:

* assertThat(object, equalTo(compareObject));
  + Метод позволяет сравнить object с compareObject, и в случаи несоответствия уронит тест, что обязан увидеть разработчик.

Вместо equalTo можно использовать большое кол-во различных Matchers, подробнее можно ознакомиться <http://hamcrest.org/JavaHamcrest/javadoc/1.3/org/hamcrest/Matchers.html>.

Так же Mockito представляет очень полезную функцию подмены значений при вызове метода у Mock объекта:

* then(mockObject.callMethod(eq(param))).thenReturn(object);
  + Метод позволяет в случаи вызова метода callMethod с параметром param во временя тестирования, подменить результат работы этой функции object.

За счёт описанных выше методов и средств были написаны тесты на все классы и методы.

После написания всех тестов в консоли можно написать mvn clean test и система запустит все тесты, которые имеются в проекте. Если всё прошло успешно, то в результате поля Failures, Errors и Skipped должны иметь значение 0, как на рисунке 2.

**2 Интеграционное тестирование**

Интеграционное тестирование производятся над прослойкой dao и внешним сервисом. Для проведения такого теста следует полностью поднимать контекст spring приложения. Для этого необходимо создать конфигурационный класс для поднятия тестового контекста. Листинг кода приведен ниже.

@Configuration  
@ComponentScan(value = "com.example.journal\_coast.dao")  
public class TestConfig implements DisposableBean {  
 private final String bindIp = "localhost";  
 private final int port;  
 private final MongodExecutable mongodExecutable;  
 private final MongodProcess mongodProcess;  
  
 TestConfig() throws IOException {  
 port = Network.*getFreeServerPort*();  
 MongodStarter starter = MongodStarter.*getDefaultInstance*();  
 try {  
 MongodConfig mongodbConfig = MongodConfig.*builder*()  
 .version(Version.*V4\_0\_12*)  
 .net(new Net(bindIp, port, Network.*localhostIsIPv6*()))  
 .build();  
 mongodExecutable = starter.prepare(mongodbConfig);  
 mongodProcess = mongodExecutable.start();  
 } catch (IOException e) {  
 destroy();  
 e.printStackTrace();  
 *log*.error("Init embedded mongo thrown exception: {}", e.getMessage(), e);  
 throw e;  
 }  
 }  
  
 @Override  
 public synchronized void destroy() {  
 if (mongodProcess != null) {  
 mongodProcess.stop();  
 }  
 if (null != mongodExecutable) {  
 mongodExecutable.stop();  
 }  
 }  
  
 @Bean  
 public MongoClient mongoClient() {  
 return MongoClients.*create*("mongodb://" + bindIp + ":" + port);  
 }  
  
 @Bean  
 public SimpleMongoClientDbFactory mongoDbFactory() {  
 return new SimpleMongoClientDbFactory(mongoClient(), "journal");  
 }  
  
 @Bean  
 public MongoTemplate mongoTemplate() {  
 return new MongoTemplate(mongoDbFactory());  
 }  
}

После создание config-файла в тестируемом классе обозначим параметры запуска с тестовой конфигурации. Исполняется при помощи аннотации, приведенной ниже.

@RunWith(SpringRunner.class)  
@SpringBootTest(classes = TestConfig.class)

После этого в коде тестируемого класса следует «привязать» @Bean тестируемого класса при помощи аннотации @Autowired. Также в силу специфики интеграционных БД, следует привязать базовый @Bean для работы с выбранной базой (в нашем случае это MongoTemplate). При помощи аннотаций @Before и @After можно добавить методы, которые будут инициализировать начальное состояние БД и очищать после работы каждого теста. Итоговый листинг одного из тестов приведен ниже.

@RunWith(SpringRunner.class)  
@SpringBootTest(classes = TestConfig.class)  
public class SubjectDaoTest {  
 @Autowired  
 private MongoTemplate mongoTemplate;  
 @Autowired  
 private SubjectDao subjectDao;  
  
 @After  
 public void clear() {  
 mongoTemplate.findAllAndRemove(new Query(), Subject.class);  
 }  
  
 private final Subject subject = Subject.*builder*()  
 .id("id")  
 .name("name")  
 .build();  
  
 @Test  
 public void getSubject() {  
 mongoTemplate.insert(subject);  
  
 Subject result = subjectDao.getSubject(subject.getId());  
  
 *assertThat*(result, *equalTo*(subject));  
 }  
  
 @Test  
 public void saveSubject\_add() {  
  
 subjectDao.saveSubject(subject);  
  
 List<Subject> result = mongoTemplate.find(new Query(), Subject.class);  
 *assertThat*(result, *hasSize*(1));  
 *assertThat*(result.get(0), *equalTo*(subject));  
 }  
  
 @Test  
 public void saveSubject\_update() {  
 subjectDao.saveSubject(subject);  
 Subject update = Subject.*builder*()  
 .id(subject.getId())  
 .name(subject.getName() + "hehe")  
 .description("desc")  
 .build();  
  
 subjectDao.saveSubject(update);  
  
 List<Subject> result = mongoTemplate.find(new Query(), Subject.class);  
 *assertThat*(result, *hasSize*(1));  
 *assertThat*(result.get(0), *equalTo*(update));  
 }  
  
 @Test  
 public void deleteSubject() {  
 subjectDao.saveSubject(subject);  
  
 subjectDao.deleteSubject(subject.getId());  
  
 List<Subject> result = mongoTemplate.find(new Query(), Subject.class);  
 *assertThat*(result, *hasSize*(0));  
 }  
}

**Вывод:** получены практические навыки создания автоматических тестов и привязки их к тестовым случаям.