БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики и информатики
Кафедра технологий программирования

Лабораторная работа № 6
По курсу “Проектирование человеко-машинных интерфейсов”

Проектирование и разработка веб-приложения и мобильного приложения. Построение окружения на основе docker-контейнеров

Методические указания по выполнению лабораторной работы

Подготовила: Давидовская М.И.,
Старший преподаватель кафедры ТП

Минск, 2022 г.

СОДЕРЖАНИЕ

Введение

В результате выполнения лабораторных работ студент должен приобрести следующие навыки и умения:

- проектирование ПО на основе человеко-центрированного подхода;
- разработка прототипов интерфейсов,
- применение принципов и шаблоны проектирования взаимодействия;
- анализ и оценка пользовательского интерфейса ПО;
- проектирование и разработка мобильного приложения (HTML5, CSS3) и веб-приложения (PHP/Python (Django)/Ruby (Ruby on Rails)/Java(Java Spring Framework: Spark, Spring MVC)/C#(Asp.Net MVC)); css-фреймворки (Bootstrap, Zurb Foundation), js- фреймворки (jQuery, AngularJS, React, Knockout, Vue) и др.

Лабораторные работы связаны между собой единой темой из предложенного преподавателем списка или выбранной студентом самостоятельно при условии предварительного согласования с преподавателем. Входными данными для каждой последующей работы являются результаты предыдущей.

Рекомендации по выполнению лабораторной работы

Цель работы — получить навыки создания окружения на основе технологии контейнеризации Docker.

Выполнение лабораторной работы № 6 состоит из следующих этапов:

2. Разработка технического задания для внутреннего использования. Создается на основе отчета из первой лабораторной работы и включает:
    - постановку задачи,
    - стратегию дизайна,
    - диаграммы бизнес-процессов и диаграммы вариантов использования,
    - диаграммы деятельности,
    - диаграммы классов и объектов,
    - диаграммы компонентов,
    - диаграммы развертывания,
    - схему базы данных, используя диаграммы "сущность-связь" (Entity RelationShip Diagram — ERD) и, при необходимости, диаграмму объектно-реляционного отображения (Object-relational mapping – ORD).
2. Изучить статью и последовательность шагов, необходимых для построения окружения разработки https://habr.com/ru/company/southbridge/blog/329262/
3. Создать структуру проекта в репозитории проекта на github. Создать ветки для релиз-версии (production) и тестовой версии (development). Согласовать правила работы с ветками. Разработку проекта вести только с использованием системы контроля версий.
4. Управление проектом вести с использованием Project в github. Для коммуникации использовать Slack или иной мессенджер для командной работы. При необходимости создать необходимое количество проектов в основном репозитории, например:
    - Проект “Дизайн и проектирование приложений” – задачи по материалам лабораторных работ 1-5
    - Проект “Разработка API и бизнес-логики”
    - Проект “Разработка веб-приложения”
    - Проект “Разработка мобильного приложения”
5. На основе технического задания, разработанного в п. 1 данной лабораторной работы, составить план работ по бекенду, включая разработку API, распределив задачи между участниками команды.
6. План работ добавить в проекты в системе контроля версий в основной репозиторий и документировать проект в Readme, wiki и Github Pages репозитория согласно требованиям, представленным в Документирование проекта.

7. Добавить автоматические тесты, чтобы сервер CI перед сборкой выполнял тесты и формировал сборку только после их прохождения.
8. Установить docker и контейнеры, требуемые для разработки приложения. Добавить простое приложение типа «Hello World» для иллюстрации работоспособности настроенного окружения.
9. Для автоматизации сборки проекта настроить сервис непрерывной интеграции (Сервис travis-CI). Обеспечить сборку с docker-контейнерами.
10. Доработать процессы CI/CD и встроить генерацию необходимых файлов для Swagger одним из этапов сборки.
11. Настроить сервис Better Code Hub (https://bettercodehub.com/) для проверки качества кода.
12. Обеспечить публикацию проекта на Heroku, OpenShift или Google Cloud.
13. Создание отчета, описывающего работы по всем пунктам данного задания и вклад каждого участника проекта, включая информацию по комитам из git-репозитория и распределение работ в рамках проекта.

Для публикации на внешнем хостинге можно использовать бесплатные облачные платформы OpenShisft, Heroku или Google Cloud. Для развертывания окружения разработки локально использовать контейнеры на примере Docker или LXC/LXD. Продемонстрировать подключение к окружению разработки и запуск приложения на облачном сервисе.

План выполнения лабораторной работы:

№	Задача
1.	Техническое задание
2.	Распределение задач на настройку окружения и назначение ответственных
3.	Создание аккаунта на Openshift/Heroku/Google Cloud
4.	Настройка окружения разработки с использованием docker.
5.	Создание автоматических тестов и настройка Travis-CI
6.	Создание схемы БД и развертывание
7.	Создание плана работ по разработке бекенда (логики), включая API, приложения.

Рекомендации по использованию инструментов разработки

Что должен уметь backend-разработчик

https://habrahabr.ru/company/netologyru/blog/328426/

https://geekbrains.ru/posts/profession_web_developer

http://bifot.ru/chto-dolzhen-umet-back-end-razrabotchik-developer-php/

https://kurspc.com.ua/node/389

https://jetbrains.ru/careers/requirements/backender/

Управление проектами

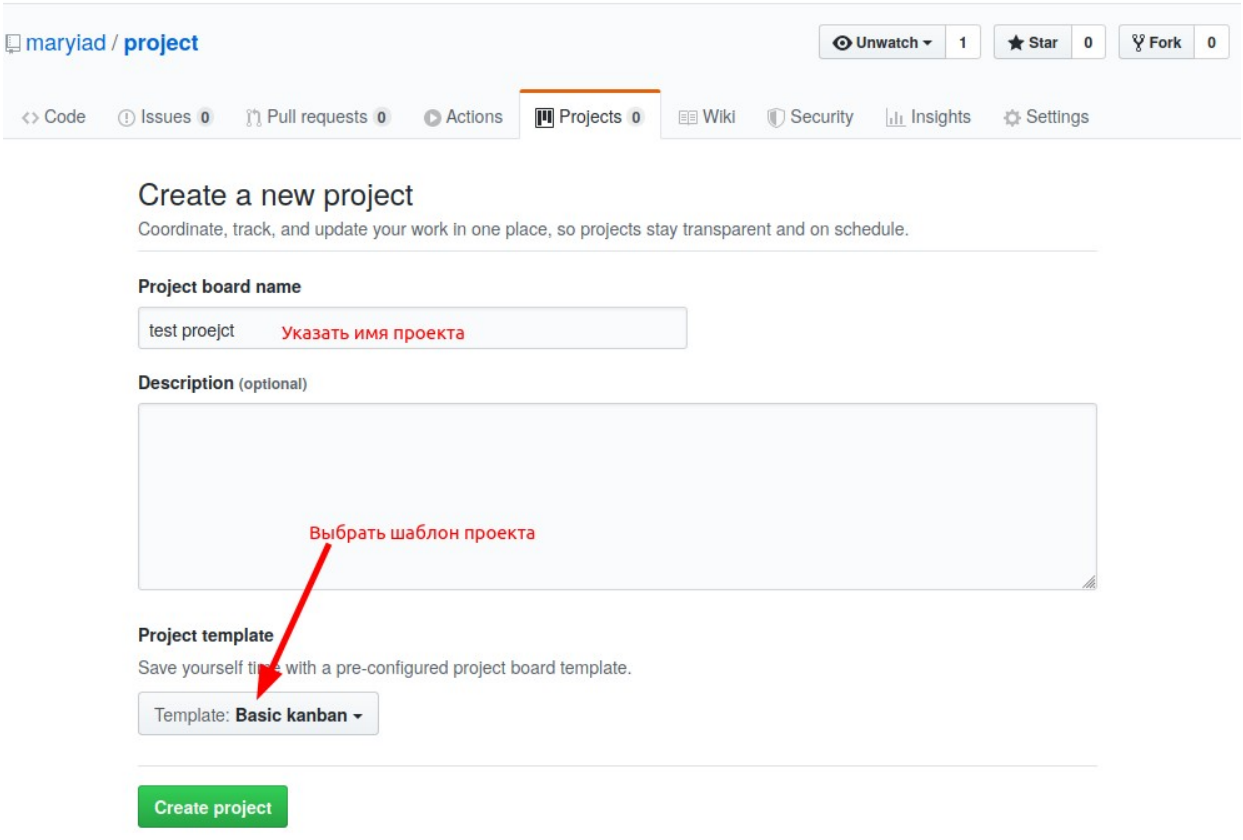
Для управления проектами применяются различные системы и решения, например Jira, Trello, Targetprocess, Github Projects и другие.

Руководства и рекомендации по системам управления проектами:

- Github project
  https://help.github.com/en/github/managing-your-work-on-github/about-project-boards
- Trello
  https://trello.com/ru/guide

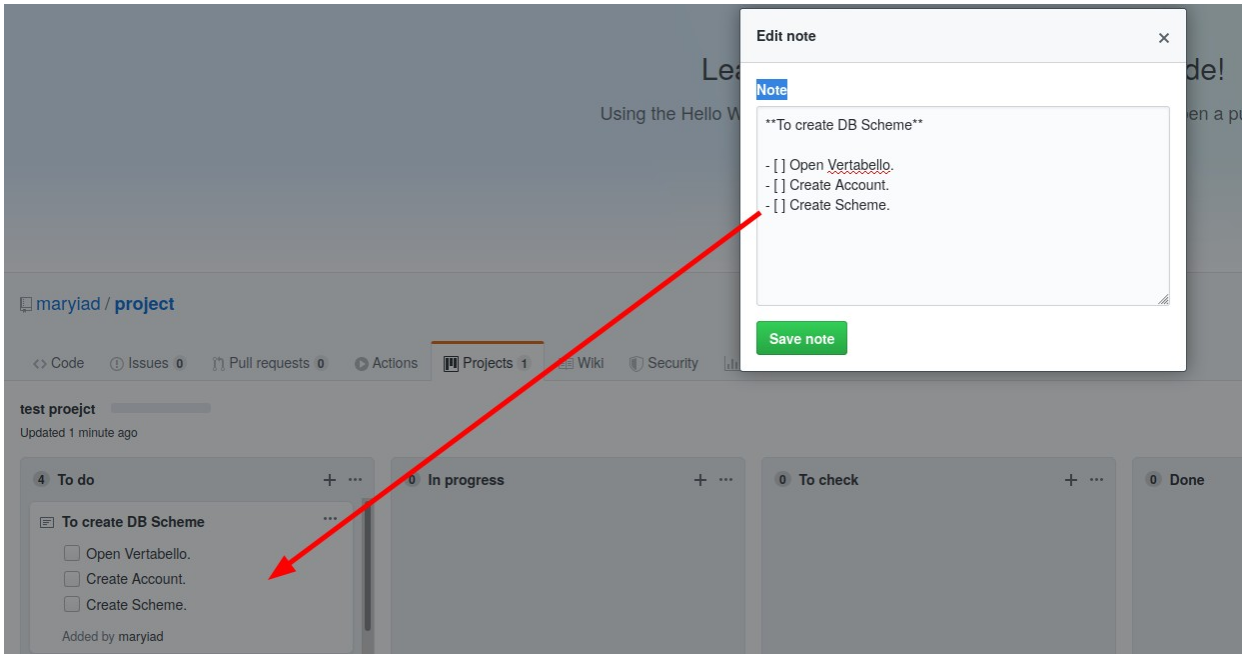
Порядок действий при создании проекта в Github Projects

1. Ознакомьтесь с документацией по управления проектами в GitHub — Managing project boards.
2. Учтите, что на основе каждой задачи в проекте необходимо создать тикет (issue). Подробнее о работе с тикетами в статье «Как эффективно работать с тикетами (issues) на GitHub» и пример организации репозитории https://github.com/DotNetRu/Server/issues.
3. Тимлид команды в репозитории создаёт Проект (см. рис. ниже):



4. К сгенерированным спискам `To do`, `In progress`, `Done` добавляет список `To check`.

5. Для добавления в список задачи нажмите символ + в заголовке списка. Например, описание задачи с чеклистом:



6. Как только задача готова, перемещается в список `To check` и передается на тестирование участнику команды, который выполняет тестирование.
7. После прохождения тестов задача перемещается в список `Done`.

Порядок действий при создании проекта в Trello

1. Ознакомьтесь с документацией https://trello.com/guide/trello-101.
2. Тимлид команды создаёт учётную запись в `Trello`.
3. Создаёт команду и отправляет приглашения другим участникам команды.
4. Создаёт доску.
5. Создаёт списки для задач, например `Новая`, `В работе`, `Тестирование`, `Готово`.
6. Добавляет задачи в список `Новая` и, по возможности, распределяет задачи между участниками команды.
7. Как только задача готова, перемещается в список `Тестирование` и передаётся на тестирование участнику команды, который выполняет тестирование.
8. После прохождения тестов задача перемещается в список `Готово`.

Документирование проекта

Документирование проекта в README wiki

Изучить документацию https://guides.github.com/features/wikis/ и документировать проект в Readme и wiki репозитория согласно следующим требованиям:

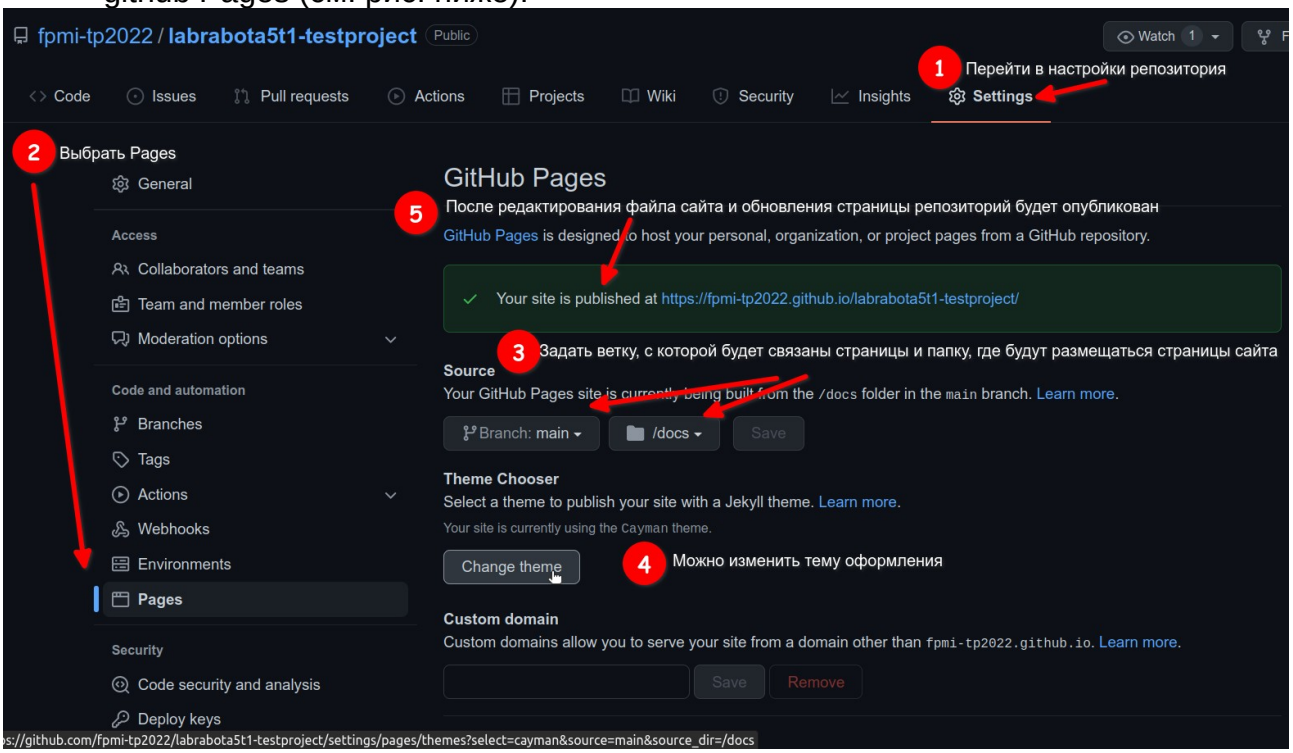
1. Файл Readme и wiki оформить с помощью синтаксиса Markdown.
2. Структура файла Readme должна быть следующей:
   - **Project Name**: в данном блоке указать название проекта.
   - **Description**: Краткое описание проекта и его функциональности в 3-5 предложениях.

- **Installation**: Последовательность шагов, как установить приложение локально.
- **Sub modules**: Ссылки на репозитории для веб-приложения, мобильного приложения с их кратким описанием.
- **Usage**: Рекомендации как использовать приложение после установки. Может содержать скриншоты.
- **Contributing**: Сведения об авторах проекта и какие задачи реализовывали.

3. Структура страниц wiki должна быть следующей:
   - **Главная страница**: содержит краткое описание задачи и ссылки на другие материалы и разделы.
   - **Функциональные требования**: описание функциональных требований, диаграммы Use case, текстовые сценарии.
   - **Диаграмма файлов приложения**: диаграмма файлов и описание.
   - **Дополнительная спецификация**: ограничения, требования к безопасности, надежности и другое.
   - **Схема базы данных**: страница содержит схему базы данных в виде изображения и ссылку на sql-файл.
   - **API-проекта**: описание API и основные запросы
   - **Презентация проекта**: ссылка на презентацию проекта, в которой должно быть отражено распределение задач в команде, требования к приложению, схема базы данных, как была организовано работа с репозиторием и проектом и др.

Описание проекта команды в Github Pages

1. Изучить курс https://lab.github.com/githubtraining/github-pages
2. В командном основном репозитории для проекта тимлид создаёт сайт для github Pages (см. рис. ниже).



Пример сайта на основе Github Pages доступен по ссылке — https://fpmi-tp2022.github.io/labrabota5t1-testproject/

3. Добавить структуру страниц в сайт Github Pages, аналогичную wiki. Оформление сайта будет учитываться при выставлении оценки за проект по

итогам курса.

Непрерывная интеграциия, настройка, юнит-тесты и другие виды тестов

Автоматизация сборки

Как создать современную CI/CD-цепочку с помощью бесплатных облачных сервисов – https://habr.com/ru/company/southbridge/blog/329262/

Создание собственного образа Docker — https://www.dmosk.ru/miniinstruktions.php?mini=docker-self-image

Использование Docker для чайников — https://losst.ru/ispolzovanie-docker-dlya-chajnikov

CI/CD в Github Actions для проекта на Flask+Angular — https://prohoster.info/blog/administrirovanie/ci-cd-v-github-actions-dlya-proekta-na-flask-angular

Инфраструктура сборки проекта с docker — https://habr.com/ru/post/457870/

Создание CI/CD-цепочки и автоматизация работы с Docker — https://temofeev.ru/info/articles/sozdanie-ci-cd-tsepochki-i-avtomatizatsiya-raboty-s-docker/

ASP.NET 5: непрерывная интеграция с Travis-CI, Tutum, Docker, Webhooks и Azure — https://coderlessons.com/articles/devops-articles/asp-net-5-nepreryvnaia-integratsiia-s-travis-ci-tutum-docker-webhooks-i-azure

http://habrahabr.ru/post/155201/

Непрерывная интеграция и функциональное тестирование: два ключевых фактора разработки качественной информационной системы - http://www.epam.by/aboutus/news-and-events/articles/2011/aboutus-ar-07-14-2011.html

QA Automation - http://andrebrov.net/blog/qa-automation-часть-вводная/

QA Automation, часть 1: CI сервер наше все - http://andrebrov.net/blog/qa-automation-часть-1-ci-сервер-наше-все/

QA Automation, часть 2: Автотестирование – Начало. - http://andrebrov.net/blog/qa-automation-часть-2-автотестирование-начало/

Travis-CI

1. https://docs.travis-ci.com/user/build-stages/share-docker-image/
2. https://docs.travis-ci.com/user/docker/
3. https://medium.com/mobileforgood/patterns-for-continuous-integration-with-docker-on-travis-ci-71857fff14c5
4. https://techblog.xavient.com/introduction-to-docker/
5. https://android.jlelse.eu/where-android-and-docker-meet-a2a34130a504
6. https://medium.com/@elye.project/intro-to-docker-building-android-app-cb7fb1b97602

Разработка API

Курс по документированию REST API — https://starkovden.github.io/.

Курс по документированию REST API — https://github.com/docops-hq/learnapidoc-ru

Как создать REST API. Простой пример — https://codeofaninja.com/2017/02/create-simple-rest-api-in-php.html

Создание REST API для проекта на Yii2 — https://klisl.com/yii2-api-rest.html

Автоматизируем документирование кода с помощью Swagger — https://nixys.ru/avtomatiziruem-dokumentirovanie-koda-s-pomoshhju-swagger/

Swagger – умная документация вашего RESTful web-API — обзор Junior back-end developer-а для новичков — https://habr.com/ru/post/434798/

Запросы

Генератор данных

https://www.mockaroo.com/

Postman

Клиент для тестирования запросов к сайтам - https://www.getpostman.com/apps

Кроме десктопных приложений есть расширение для браузера Chrome. Для Firefox можно использовать другие расширения, например HttPrequest

Использование postman для тестирования запросов -

https://mindbox.fogbugz.com/default.asp?W1299

Настройка окружения с Docker

Принцип работы контейнеров можно легко понять использовав концепт виртуальных машин. Подобно виртуальным машинам, контейнеры обеспечивают безопасность путем одновременного запуска отдельных экземпляров операционных систем без взаимодействия друг с другом. Кроме того, как и виртуальные машины, контейнеры повышают мобильность и гибкость ваших проектов, так как вы не зависимы от какого-либо конкретного оборудования и можете перейти на любое другое облако, локальную среду и т.д.

Но в отличие от виртуальных машин, для которых требуется установка полнофункциональных операционных систем, что вызывает дополнительную нагрузку. Контейнеры совместно используют ядро одной операционной системы, сохраняя при этом изоляцию по отношению к другим контейнерам. Проще говоря, вы получаете тот же результат, что и при использовании виртуальных машин, но без дополнительной нагрузки.

Docker использует такую же схему для создания контейнеров на VM на базе Linux. В одном контейнере Docker, вы получаете доступ ко всем необходимым вам ресурсам: исходному коду, зависимостям и среде выполнения.

1. https://habr.com/ru/company/southbridge/blog/329262/
2. https://atlogex.com/docker-start/
3. https://otus.ru/events/devopsopen/84/
4. https://habr.com/ru/company/ruvds/blog/438796/
5. https://www.hostinger.ru/rukovodstva/kak-ustanovit-wordpress-na-docker#--Docker
6. https://blog.amartynov.ru/docker-%D0%BD%D0%B0-windows-%D1%83%D0%B6%D0%B5-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%B5%D1%82/
7. https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-docker/configure-docker-daemon

Развертывание docker-образа на облачной платформе

1. https://blog.machinebox.io/deploy-docker-containers-in-google-cloud-platform-4b921c77476b
2. https://medium.com/google-cloud/deploying-docker-images-to-google-cloud-using-kubernetes-engine-637af009e594
3. https://scotch.io/tutorials/google-cloud-platform-i-deploy-a-docker-app-to-google-container-engine-with-kubernetes
4. https://blog.openshift.com/deploying-applications-from-images-in-openshift-part-one-web-console/
5. https://devcenter.heroku.com/categories/deploying-with-docker
6. https://cloud.google.com/cloud-build/docs/quickstart-docker
7. https://cloud.google.com/run/docs/deploying
8. https://documentation.codeship.com/pro/continuous-deployment/google-cloud/

Сервисы проверки качества кода

https://bettercodehub.com

https://www.reshiftsecurity.com

https://www.sonarqube.org

Сервисы временной почты

https://temp-mail.org/

https://temp-mail.ru/

https://dropmail.me/ru/

Другие полезности

Pastebin

Pastebin is a website where you can store any text online for easy sharing. The website is

mainly used by programmers to store pieces of sources code or configuration information, but anyone is more than welcome to paste any type of text. The idea behind the site is to make it more convenient for people to share large amounts of text online.
https://pastebin.com/