Разработчик C# + .NET

Андрей Голяков

Преподаватель курса

Андрей Голяков

Обладаю 15-летним опытом работы в сфере IT и программирования на языках C#, SQL, JavaScript.

С 2006 работаю в компании CNET Content Solutions – филиале компании CBS Interactive, входящей в международный медиа-холдинг ViacomCBS.
Прошёл карьерный путь в этой компании от разработчика до руководителя сектора разработки.

Участвовал в десятке крупных проектов. Сейчас управляю несколькими командами по разработке корпоративных продуктов.

Слежу за развитием методик и инструментов для разработки и управления циклом разработки программных решений, посещаю профильные конференции.

Особенности курса

- 4 месяца (36 занятий или 180 часов)
- Обучение инструментам и методам разработки **с нуля**
- Широкий спектр **практик** для решения различных задач
- Каждые два месяца завершаются **выполнением проекта**
- Сертификат и **уверенность** при поиске работы

Доступ к материалом курса

1. Все занятия записываются
2. Запись, исходный код и презентация будут выкладываться **навечно** в ваш личный кабинет, LMS
3. Доступ будет отправлен вам по **email**

Как сдавать домашнее задание?

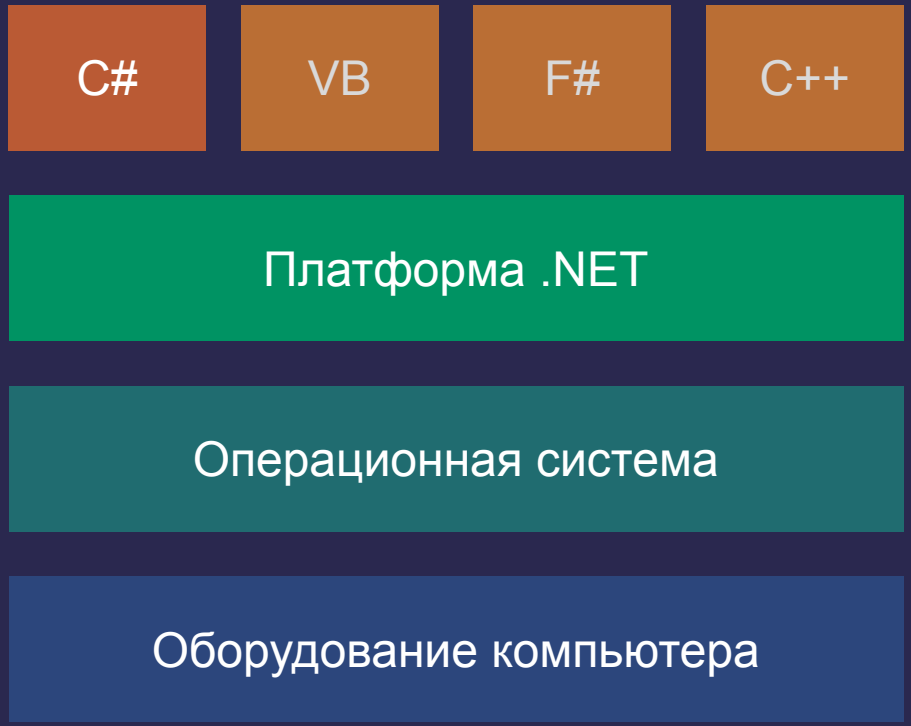
1. Домашнее задание необходимо будет загружать в LMS
2. Ваше домашнее задание будет проходить Code Review и высылаться обратно с комментариями преподавателей курса
3. Все домашние задания оцениваются, итоговый балл будет у вас в Сертификате
4. Балл можно исправить, переделав работу с учетом комментариев преподавателя

Язык C# и платформы .NET

Как выполняется код?

1. Мы пишем код на **языке C#** и запускаем его на выполнение,
2. Он преобразуется в промежуточный код платформы .NET, **общий** для всех языков верхнего уровня: **IL** (или CIL: Common Intermediate Language),
3. Платформа .NET запускает на выполнение IL-код средствами, предоставленными операционной системой,
4. Операционная система запускает код на выполнение средствами оборудования компьютера.

Язык C# и платформы .NET



Язык C# и платформы .NET

.NET Framework

- Возможность использовать специфические функции ОС Windows, однако из-за этого может разворачиваться только на Windows;
- Монолитный компонент с длительным циклом обновления;
- Код доступен только для просмотра.

.NET Core

- Кроссплатформенность: возможность запускать приложения на Windows, Mac и Linux ОС;
- Модульный компонент, что означает гибкое развёртывание и более частые обновления;
- Открытый исходный код

Средства разработки

Visual Studio Code

- Есть версии для всех ОС Windows Linux and Mac,
- Удобство однообразия среды разработки

Microsoft Visual Studio

- Имеет большое количество удобных функций для разработки приложений
- Работает на ОС Windows

Этапы разработки ПО

1. Постановка задачи
2. Проектирование
3. Кодирование
4. Отладка / тестирование
5. Сопровождение

Постановка задачи

Написать программу приветствия пользователя

- Необходимо узнать имя, подождать 5 секунд, а затем вывести ему приветствие.
- Программа должна завершиться, когда пользователь подтвердит прочтение приветствия нажатием любой клавиши на клавиатуре.

Проектирование

- Необходимо узнать имя, а потом вывести ему приветствие.
- Перед вводом приветствия должно пройти 5 секунд.
- Программа должна завершиться, когда пользователь подтвердит прочтение приветствия нажатием любой клавиши на клавиатуре.

Элементы алгоритмических блок-схем

Терминатор

(как правило, начало и конец алгоритма)

Процесс

(обработка данных, операция или группа операций)

Ввод / Вывод

(общее обозначение ввода или вывода данных)

Проектирование

- Необходимо узнать имя, а потом вывести ему приветствие.
- Перед вводом приветствия должно пройти 5 секунд
- Программа должна завершиться, когда пользователь подтвердит прочтение приветствия нажатием любой клавиши на клавиатуре.

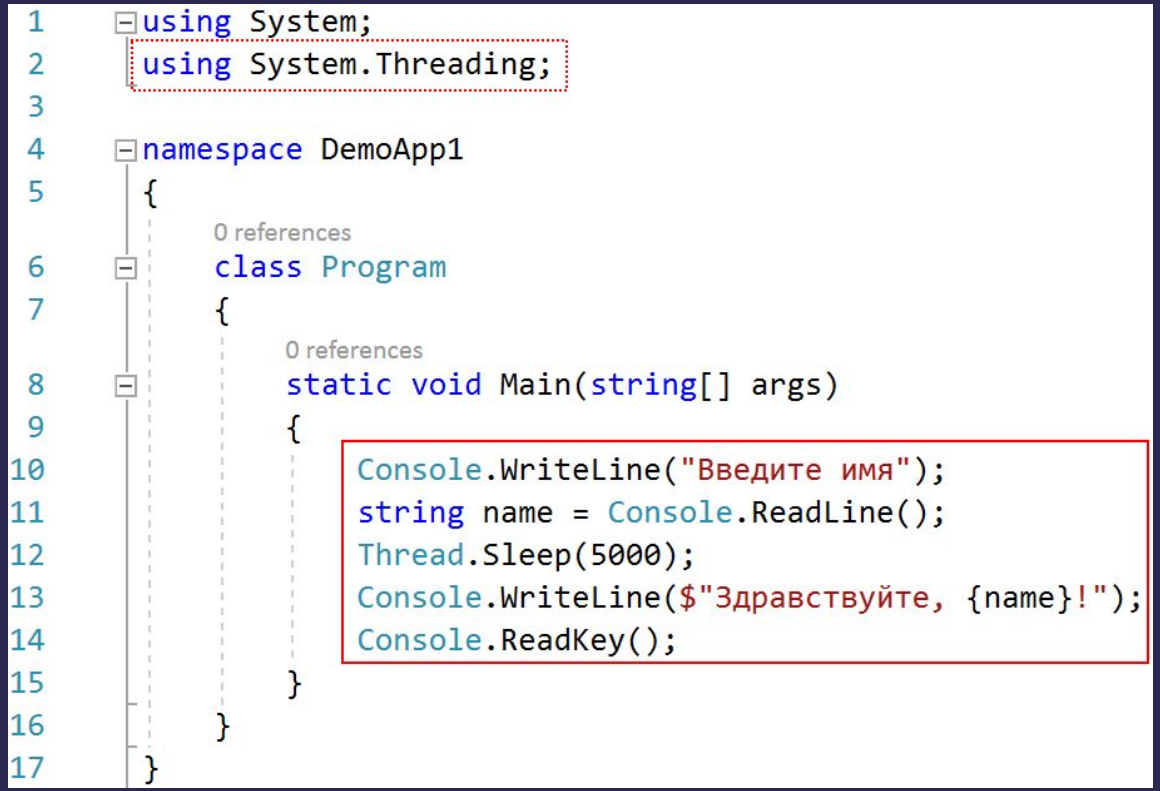
- Спросить имя
- Прочитать имя и “запомнить” его
- Подождать 5 секунд
- Вывести приветствие
- Дождаться нажатия любой клавиши

Проектирование

- Спросить имя
- Прочитать имя и “запомнить” его
- Подождать 5 секунд
- Вывести приветствие
- Дождаться нажатия любой клавиши

Кодирование

- Спросить имя
- Прочитать имя и “запомнить” его
- Подождать 5 секунд
- Вывести приветствие
- Дождаться нажатия любой клавиши



```
1   using System;
2   using System.Threading;
3
4   namespace DemoApp1
5   {
        0 references
6       class Program
7       {
            0 references
8           static void Main(string[] args)
9           {
10              Console.WriteLine("Введите имя");
11              string name = Console.ReadLine();
12              Thread.Sleep(5000);
13              Console.WriteLine($"Здравствуйте, {name}!");
14              Console.ReadKey();
15          }
16      }
17  }
```

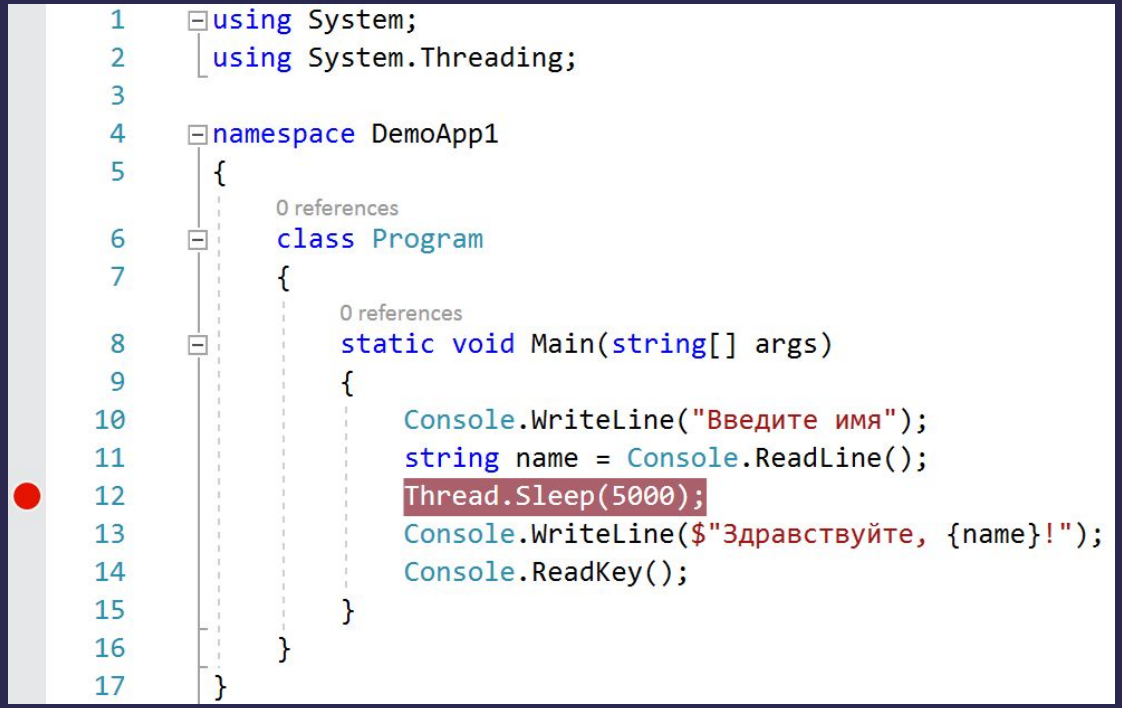
Отладка / Тестирование

Ставим точку остановки в строке номер 12

- **F9** - поставить / убрать точку остановки в режиме редактирования кода

Запуск

- **F5** - запуск в режиме отладки



Сопровождение (домашняя работа)

Модифицировать программу таким образом, чтобы после вывода приветствия программа ожидала ещё 5 секунд и выводила прощание, а уже потом ожидала нажатия клавиши и завершалась.

Версионирование исходного кода Git

GitHub

- Зарегистрироваться https://github.com

Git for Windows

- Скачать https://git-scm.com/downloads или https://gitforwindows.org
- Подробная справка по Git https://git-scm.com/doc

Основные команды Git

git clone - создать локальный репозиторий на основе внешнего

git init - создать пустой локальный репозиторий

git status - вывести статус файлов в репозитории

git add - пометить файлы для последующей операции commit

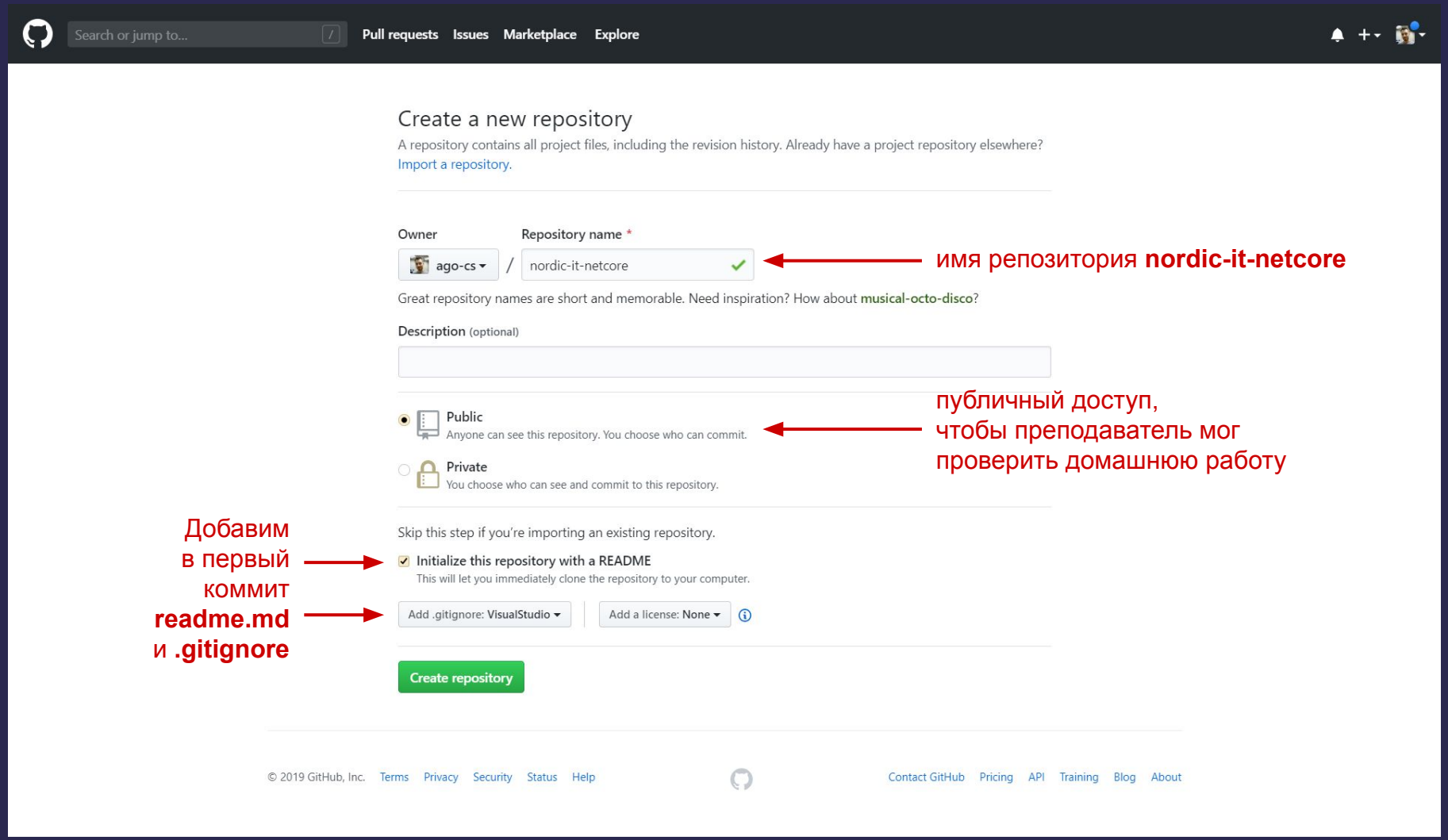
git commit - зафиксировать версию репозитория

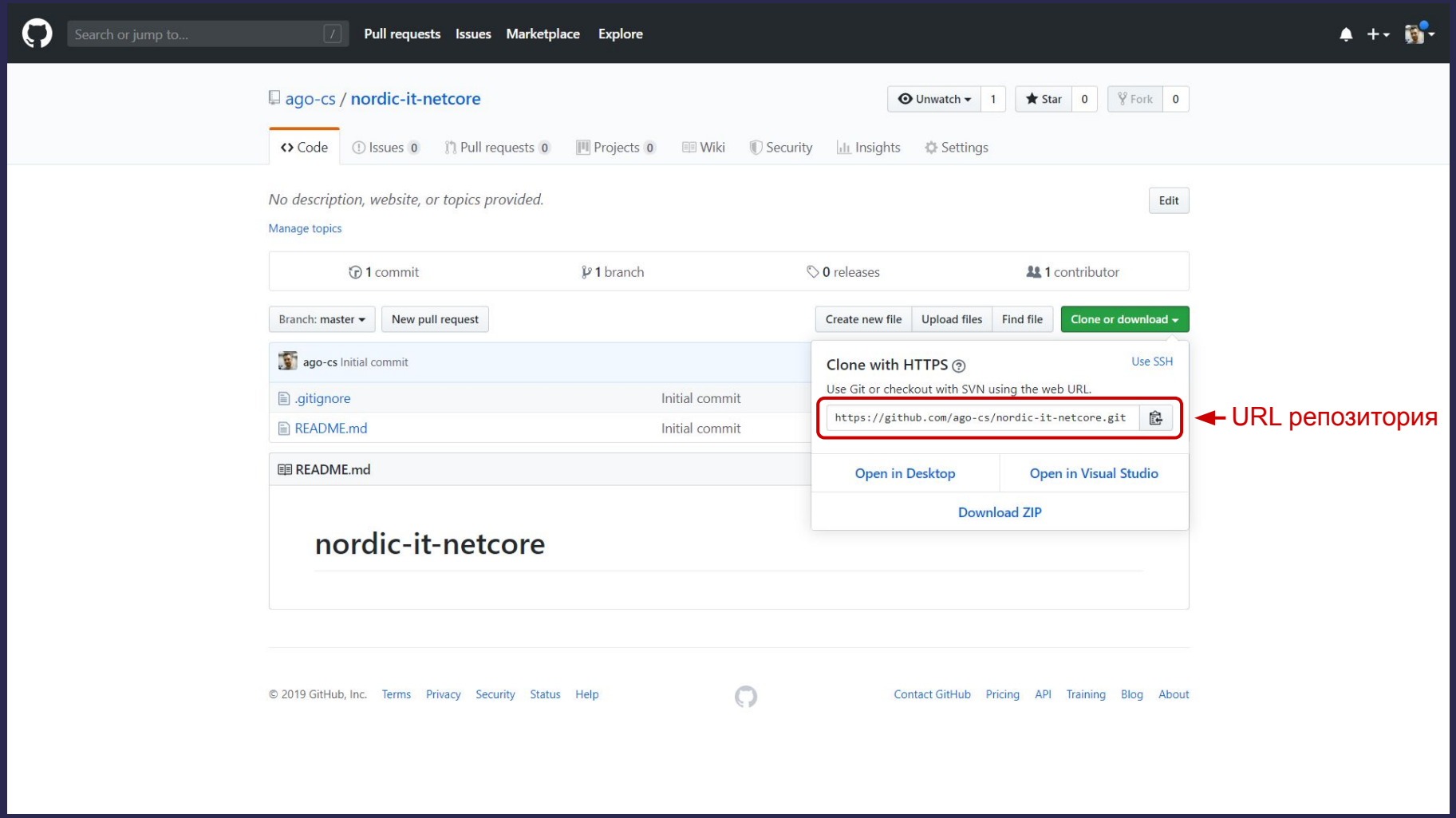
git push - отправить локальные коммиты на удалённый сервер (репозиторий)

git pull - получить изменения с удаленного репозитория

Подробная инструкция по работе с GitHub

- Регистрируемся на GitHub https://github.com
- Создаем публичный репозиторий nordic-it-netcore
  (смотри следующий слайд)

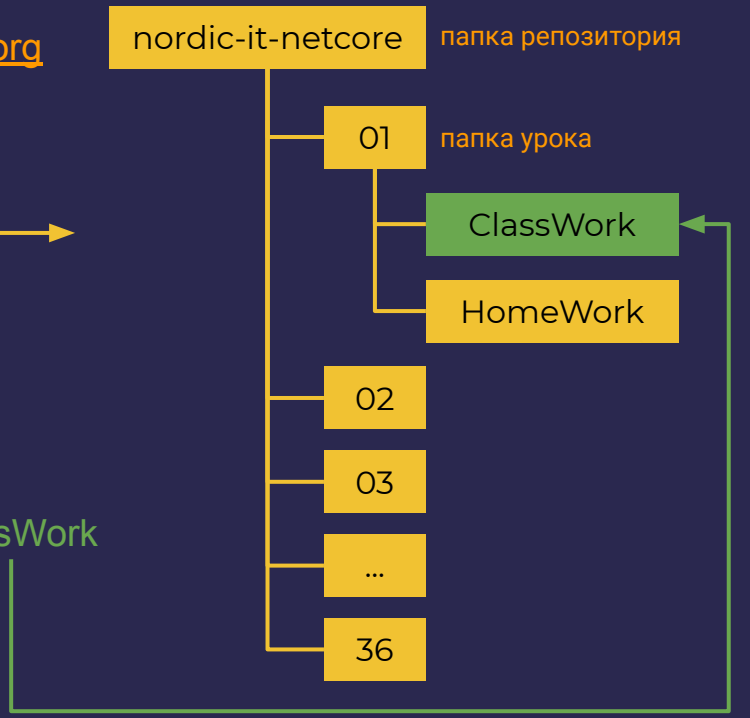

Search or jump to...
Pull requests
Issues
Marketplace
Explore
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Owner
Repository name *
ago-cs
/
nordic-it-netcore
имя репозитория nordic-it-netcore
Great repository names are short and memorable. Need inspiration? How about musical-octo-disco?
Description (optional)
Public
Anyone can see this repository. You choose who can commit.
публичный доступ,
чтобы преподаватель мог
проверить домашнюю работу
Private
You choose who can see and commit to this repository.
Skip this step if you're importing an existing repository.
Добавим
в первый
коммит
readme.md
и .gitignore
Initialize this repository with a README
This will let you immediately clone the repository to your computer.
Add .gitignore: VisualStudio
Add a license: None
Create repository
© 2019 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About



URL репозитория

Подробная инструкция по работе с GitHub

- Ставим локальный Git для Windows https://gitforwindows.org
- Запускаем Git Bash
- git clone URL-репозитория.git
- Создаем структуру папок для уроков
  - 01 (02, 03, и т.д.)
    - ClassWork
    - HomeWork
- Можно обновить содержимое файла .gitignore отсюда: https://github.com/github/gitignore/blob/master/VisualStudio.gitignore
- Переносим файлы урока в папку nordic-it-netcore/01/ClassWork



Подробная инструкция по работе с GitHub

Когда все файлы скопированы, запускаем в Git Bash последовательно следующие команды:

- git status — видим, что появились новые файлы (отображаются красным)
- git add --all — добавляем к будущему коммиту все новые файлы
- git status — файлы классной работы стали зелёными, теперь они попадут в коммит
- git commit -m "Class work of the 1st lesson added" — коммитим локально
- git push — отправляем информацию о локальном коммите на удалённый сервер

Инструкция по работе с GitHub дома

- Создаём папку, в которой будет храниться наш локальный репозиторий, например, C:\git
- Запускаем Git Bash
- Переходим во вновь созданную папку (выполняем команду cd c: , затем cd git)
- Выполняем git clone https://github.com/ваш-аккаунт/nordic-it-netcore.git,
  появится папка nordic-it-netcore с файлами классной работы первого урока
- Заходим в папку 01, создаём в ней папку HomeWork
- Запускаем VisualStudio, создаём новое приложение .NET Core, в качестве пути к проекту пишем c:\git\nordic-it-netcore\01\HomeWork, имя проекта L01_HomeWork
- Когда закончим выполняем следующие команды в Git Bash:
  - git add --all
  - git status
  - git commit -m "Home work of the 1st lesson added"
  - git push
- В личном кабинете присылаете мне ссылку на на свой GitHub-репозиторий, комментарии и вопросы.
- Если что-то не получится, присылайте вопросы, в крайнем случае просто zip-файл с кодом.

Возможные проблемы и их решение

Если на компьютере уже есть закешированные данные другой учётной записи GitHub необходимо:

- Открыть “Диспетчер Учётных Данных” (Credential Manager) > Учётные данные Windows (Windows Credentials) > Generic Credentials > git:https://github.com > Удалить эту запись.
- В следующий раз при попытке выполнить git pull снова появится диалоговое окно ввода логина/пароля учётной записи GitHub.

При первой настройке аккаунта возможна ошибка *** Please tell more who you are. Необходимо указать email и имя пользователя:

- `git config --global user.email "your@email.com"`
- `git config --global user.name "Your Name"`

Спасибо за внимание.