Частное учреждение профессионального образования

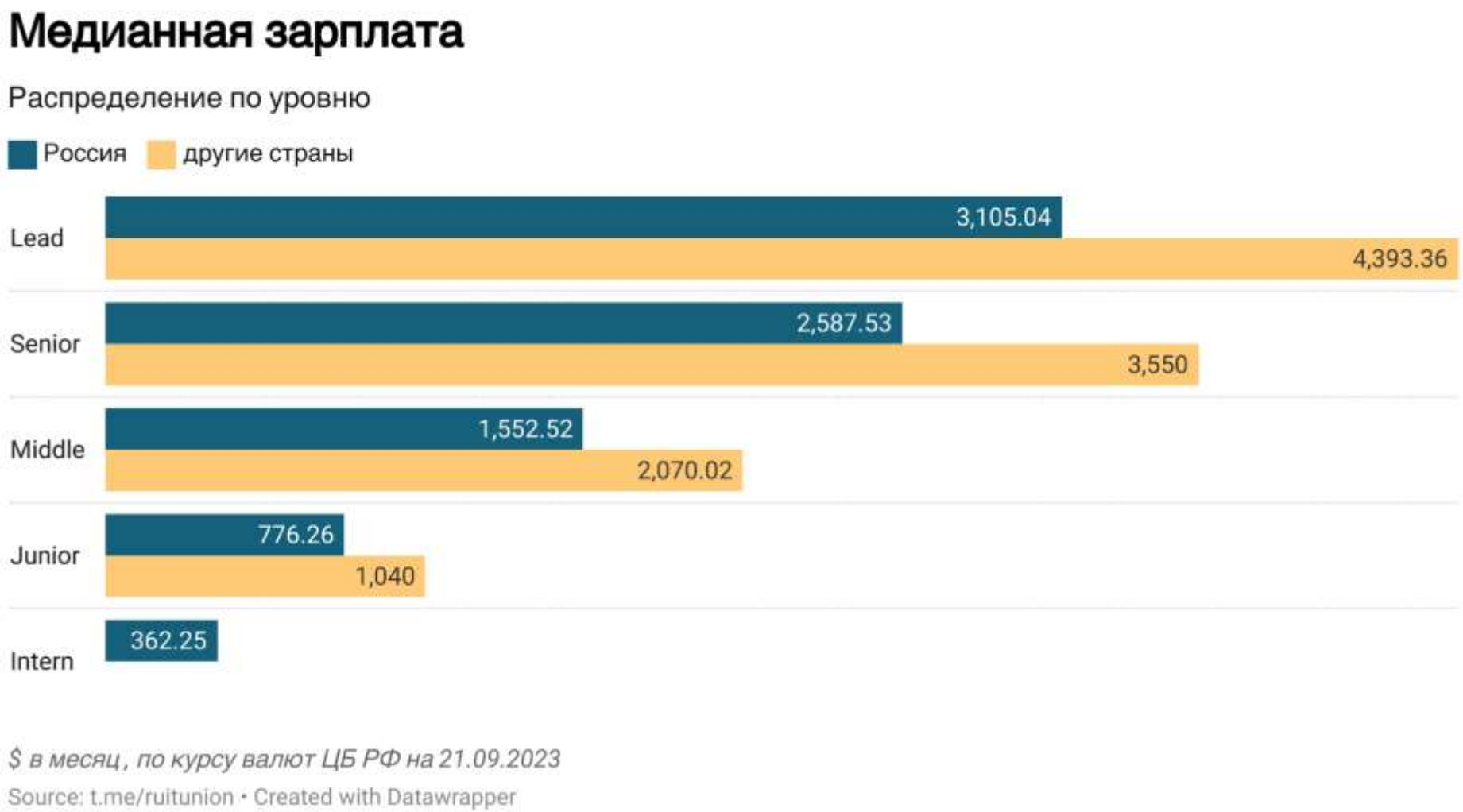
«Высшая школа предпринимательства (колледж)»

(ЧУПО «ВШП»)

Кафедра информационных технологий

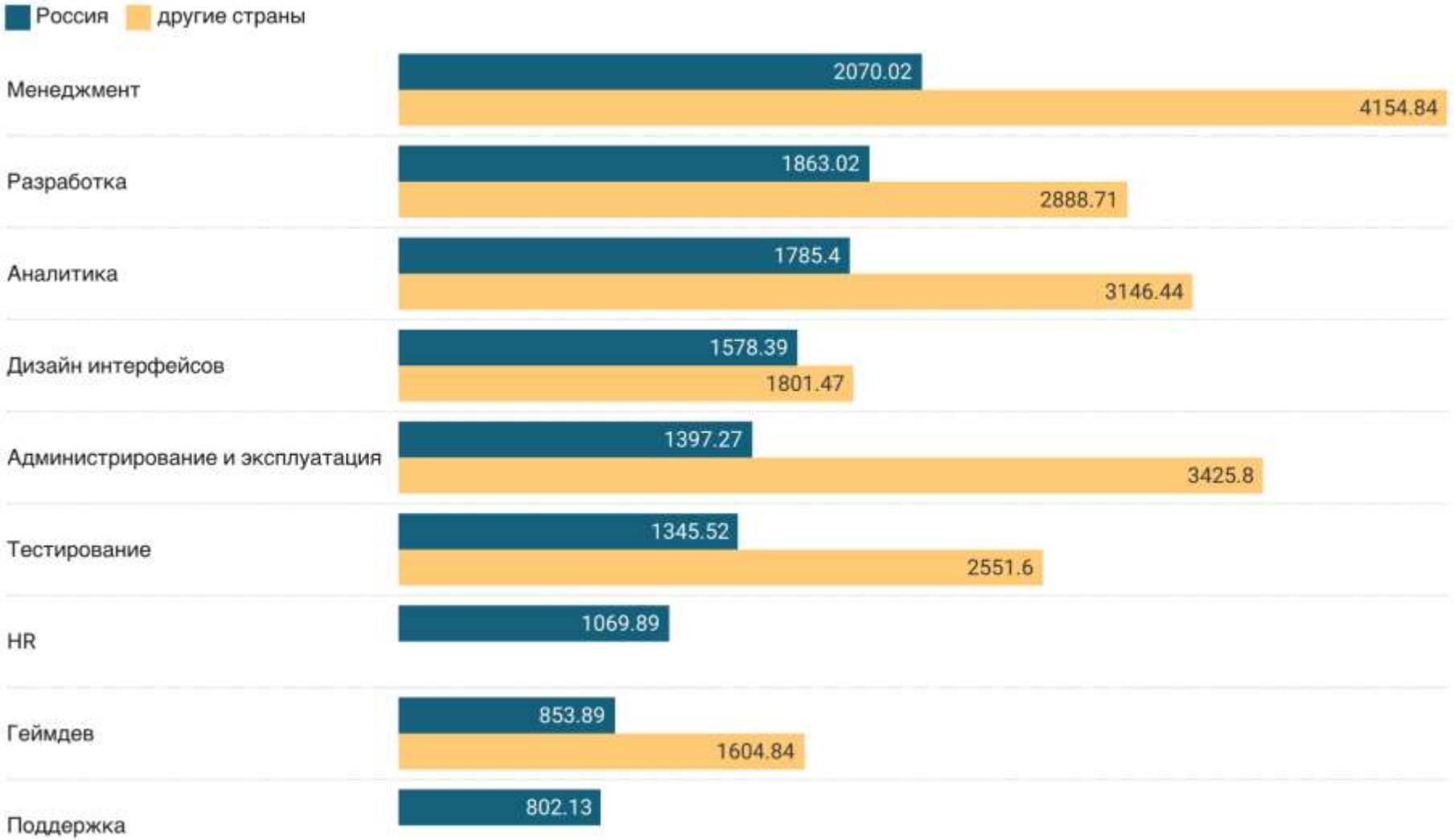
Инструментальные средства разработки программного обеспечения

Для разрядки и настройки



Медианная зарплата

Распределение по специализации



$ в месяц, по курсу валют ЦБ РФ на 21.09.2023

Source: t.me/ruitunion • Created with Datawrapper

1. **Среди опрошенных IT-специалистов медианная зарплата в России составляет $1656.02, а в других странах $2781.72.**

2. **В разрезе по уровню (Lead, Senior, Middle, Junior) медианные зарплаты айтишников в России на 25-30% меньше зарплат коллег в других странах.**

3. **Больше всех зарабатывают менеджеры, разработчики, аналитики и администраторы. Меньше всех — техподдержка, HR и работающие в геймдеве.**

4. **Только 52% айтишников в России работает полностью удалённо. Ещё 25% работает в гибридном формате. В других странах 73% и 19% специалистов работает полностью удалённо и в гибридном формате соответственно.**

5. **В России зарплата специалистов, работающих в IT-компаниях, выше на 18%, чем у специалистов IT-отделов компаний, не занятых в IT. Ситуация в других странах несколько иная — сотрудники неайтишных компаний зарабатывают на 13% больше, чем сотрудники IT-компаний.**

Что такое «система контроля версий»?

Повторим основные команды Git

git init
git status
git add
git commit
git log

git branch

git checkout

git merge

git checkout -b new-feature

git add <file>

git commit -m "feat: start a feature"

git add <file>

git commit -m "feat: finish a feature"

git checkout main

git merge new-feature

git branch -d new-feature

Требования к именам коммитов

init: - используется для начала проекта/таска.
init: start task

feat: - это реализованная новая функциональность из технического задания.
feat: add basic page
feat: add button

fix: - исправил ошибку в ранее реализованной функциональности.
fix: change layout to fix bugs

Командная строка
Основные команды

cd название_папки

dir

tree

touch index.html

mkdir project

ipconfig

GitHub

1. Установите git (например, от сюда **git-scm.com**).
2. Создайте репозиторий.
3. Создайте в репозитории текстовый файл с текстом (2-3 абзаца текста).
4. Разбейте его создание на несколько этапов (добавление 2 абзаца). Коммитьте изменения (2-3 коммита)
5. Создайте две ветки.
6. В первой ветке «поработайте» над текстом (добавьте 2 абзаца).
7. Во второй уберите содержания (удалите 1 абзац)
8. Не забывайте разбивать эту «работу» на коммиты.
9. Слейте все изменения назад в ветку master.
10. Удалите вспомогательные ветки.
11. Изучите команду git log
12. Сделайте скриншот вывод команды git log, который подтвердил бы, что Вы сделали пункты 1-10
13. Сделайте архив из папки репозитория и принесите на занятие

Срок успешной сдачи ДЗ-1

26-10-2023
23:59:59

после дедлайна задания принимаются без оценки

Markdown

Markdown — язык разметки текстов.
Такие тексты легко писать и читать. Их можно без труда сконвертировать в HTML. Используют для написания документации, описаний своих проектов и написания блогов.

«Язык разметки» — это просто набор соглашений, правил.

Допустим, что вы общаетесь с другом по СМС. В них нельзя сделать текст жирным или наклонным. Вы договариваетесь с другом: если я пишу *что-то* вот так между звездочками, то считай, что это наклонный текст. А если я пишу **что-то** между двумя звездочками, то считай, что это жирный текст. Вы придумали правила.
Markdown — это набор подобных правил.

Синтаксис Markdown

Выделение текста

Этот текст будет наклонным (курсив)
*_Этот текст будет наклонным (курсив)_*

****Этот текст будет жирным****
__Этот текст будет жирным__

*_Можно *****вставлять****** один тип в другой_*

Синтаксис Markdown

Заголовки

Это самый крупный заголовок, он превращается в тег <h1>
<h2>
<h3>
<h4>
<h5>
<h6>

Синтаксис Markdown

Ссылки

https://vshp.online/ — текст простой ссылки станет кликабельной ссылкой автоматически

Ссылкой можно сделать любой текст:

[Высшая школа предпринимательства](https://vshp.online/)

Синтаксис Markdown

Цитата

> Это мудрая цитата
> Мудрого человека

Картинки

```
![alt-текст](/assets/images/markdown/markdown.png)
```

Синтаксис Markdown

Код

Кусок кода: **`function(12);`**

Целый блок кода:

```
```javascript
const func = (num) => {
  if (num > 0) {
    return num - 1;
  }
  return num + 1;
};
```
```

Синтаксис Markdown

Списки

Непронумерованный список:

* **Пункт**
* **Еще один пункт**
  * **Подпункт**
  * **Еще один подпункт**

Пронумерованный список:

1. **Пункт**
1. **Еще один пункт**
   1. **Подпункт**
   1. **Еще один подпункт**

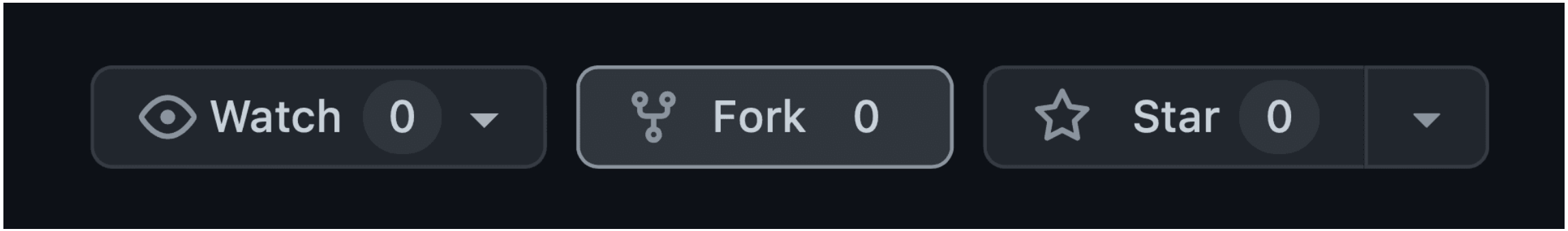
В пронумерованном списке можно использовать любые числа .

Типичная работа с Гитхабом

1. Форк оригинального удаленного репозитория
2. Клонируем локально форк
3. Создаем свою ветку для работы
4. Работаем, коммитим
5. ***Подтягиваем изменения из оригинального удаленного репозитория в локальный репозиторий***
6. Пушим изменения в форк
7. Делаем пуллреквест из форка в оригинал
8. Подтягиваем изменения из оригинального удаленного репозитория в локальный репозиторий

Форк оригинального удаленного репозитория

Заходим в нужный репозиторий и нажимаем на «вилку» с надписью **fork**



Появится окно **Create a new fork** — проверьте, что он называется так, как вам нужно, и жмите кнопку **Create fork**. Через пару секунд всё готово.

Клонируем локально форк

Клонировать форк — значит скачать его, чтобы работать с кодом на своём компьютере. Тут нам и пригодится **SSH**.

Открываем терминал в папке с будущим проектом.
В папке введите команду **git clone** для клонирования репозитория:

git clone git@github.com:your-nickname/your-project.git

Замените your-nickname на ваше имя пользователя на GitHub, а your-project на название проекта. **Проще всего их найти прямо наверху страницы репозитория.**

Если вы правильно настроили SSH-ключи, Git скопирует репозиторий на ваш компьютер.

Создаем свою ветку для работы

В терминале введите команду:

git branch

Если мы находимся в master , то создаём новую ветку командой

git checkout -b имя-новой-ветки

Мы делаем это, чтобы новая ветка содержала свежую рабочую версию проекта. Если вы ошиблись в названии, например, допустили опечатку, вы можете изменить название ветки с помощью команды:

git branch -m старое-имя-ветки новое-имя-ветки

Работаем

То, что разработчики решают задачи каждый на своей ветке, пришло к нам от наших предков — обезьян.

Пушим изменения в форк

Сохранённые изменения пока не видны никому, потому что находятся в нашем локальном репозитории. Нужно отправить коммиты на GitHub. Для этого введите команду:

git push origin название-текущей-ветки

Где **origin** означает репозиторий на компьютере, то есть ваш форк.

Слово **origin** — часть команды, не меняйте это название на своё.

Делаем пуллреквест из форка в оригинал

Пулреквест (или PR) — это предложение изменить код в репозитории. PR должен проверить администратор мастер-репозитория — это может быть коллега-разработчик, техлид.

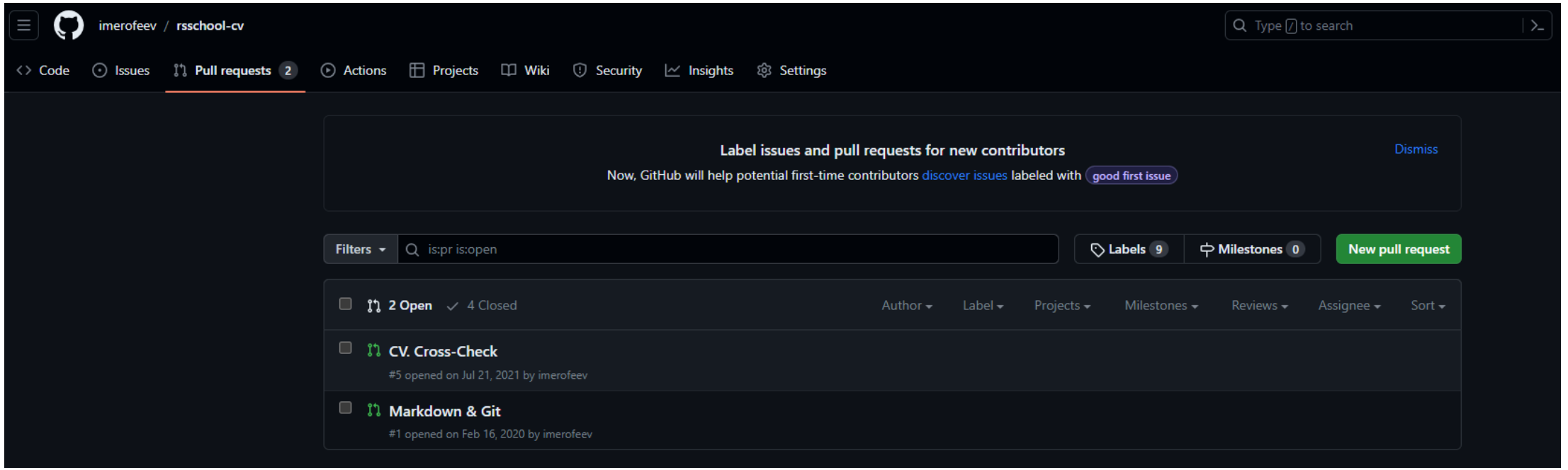
Если к коду нет вопросов, пулреквест принимается.

Если нужно что-то исправить — отклоняется, и придётся исправить код и снова пройти цепочку g**it add — git commit — git push**.

Если вы и дальше работаете в той же ветке, а пулреквест ещё не принят, все ваши изменения автоматически добавятся в пулреквест, созданный из этой ветки после команды **git push origin название-текущей-ветки**.

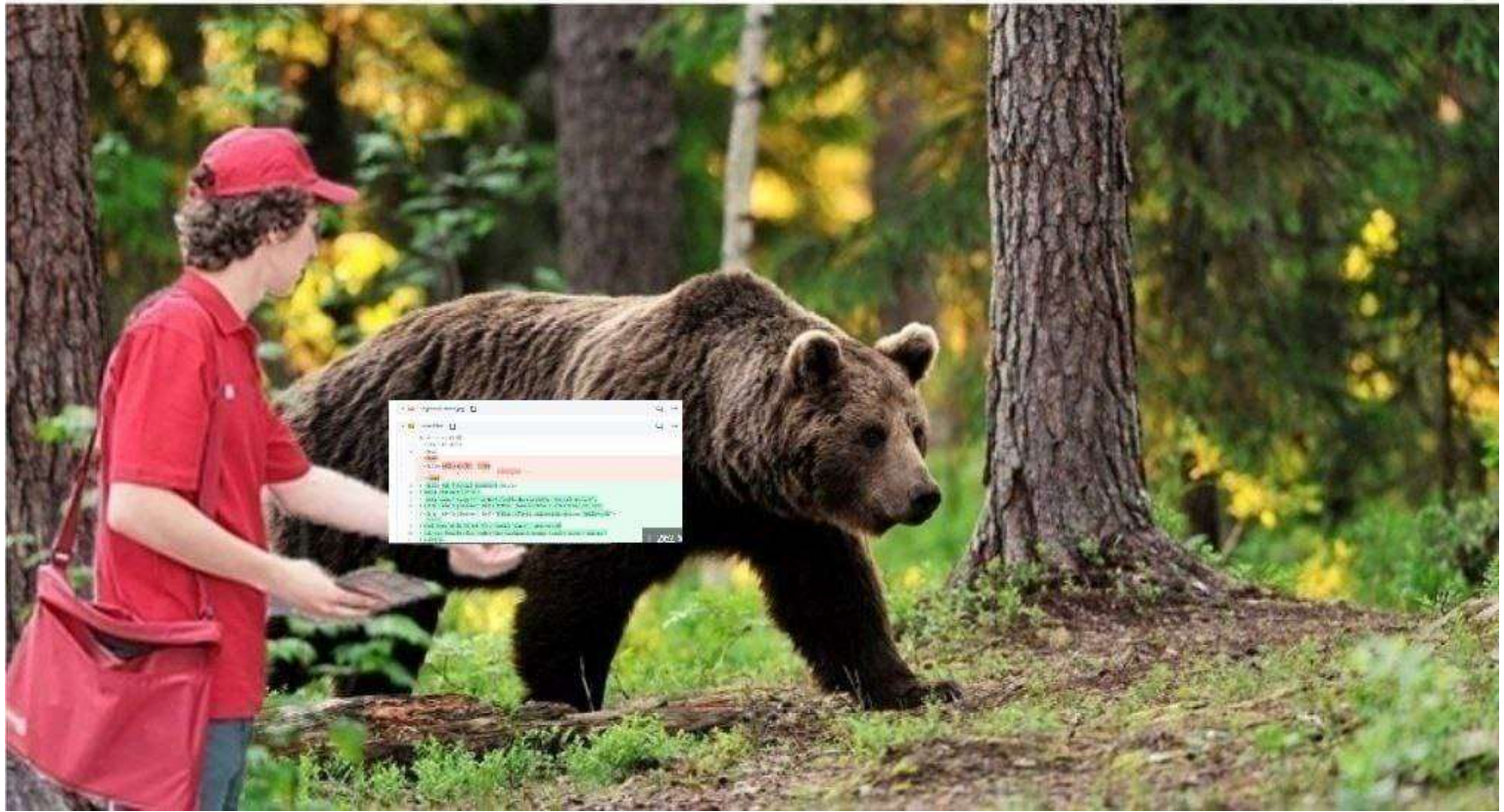
Делаем пуллреквест из форка в оригинал

Чтобы создать пулреквест, зайдите на страницу вашего форка на GitHub.
Вверху появилась плашка **Compare & pull request**, а ещё можно зайти на вкладку **Pull Requests**.
Нажмите на неё и окажетесь на странице открытия пулреквеста.
Проверьте описание и нажмите **Create pull request**.



Делаем пуллреквест из форка в оригинал

Встретив в лесу медведя, попросите его посмотреть pull request . Он сделает вид, что не заметил вас и пройдёт мимо.



Подтягиваем изменения из оригинального удаленного репозитория в локальный репозиторий

Теперь код в мастер-репозитории обновился, а в вашем форке нет, вы ведь не обновляли свою версию репозитория с тех пор, как клонировали её себе на компьютер. Приведём форк в актуальное состояние.

В локальном репозитории переключаемся на ветку **master**.

Забираем изменения из ветки **master** мастер-репозитория

git pull **git@github.com:master-repo/master-project.git**

Отправляем изменения уже из своей ветки **master** в ваш форк на GitHub с помощью команды

git push origin master

Дополнительный материал

docs.github.com/ru