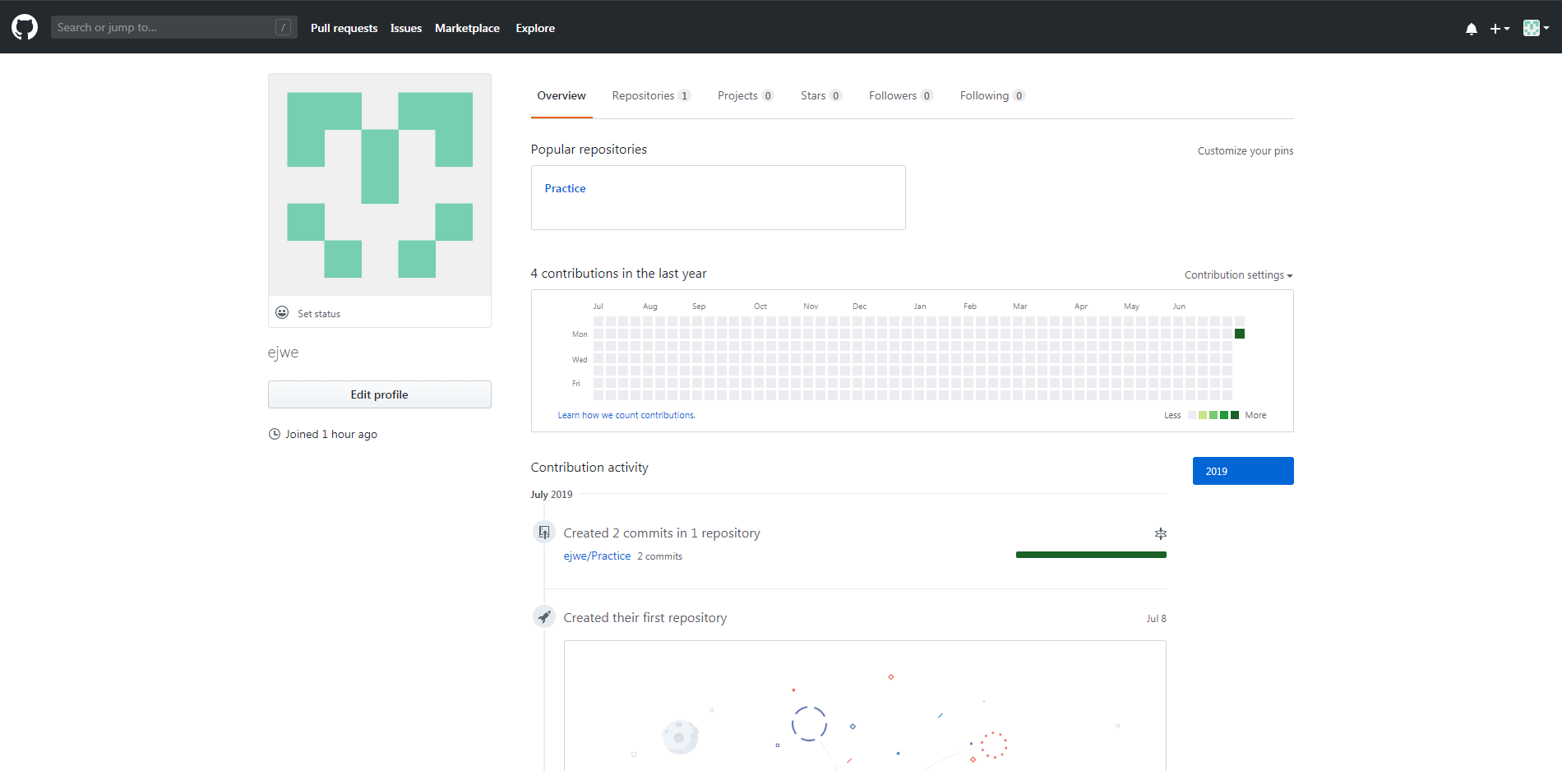
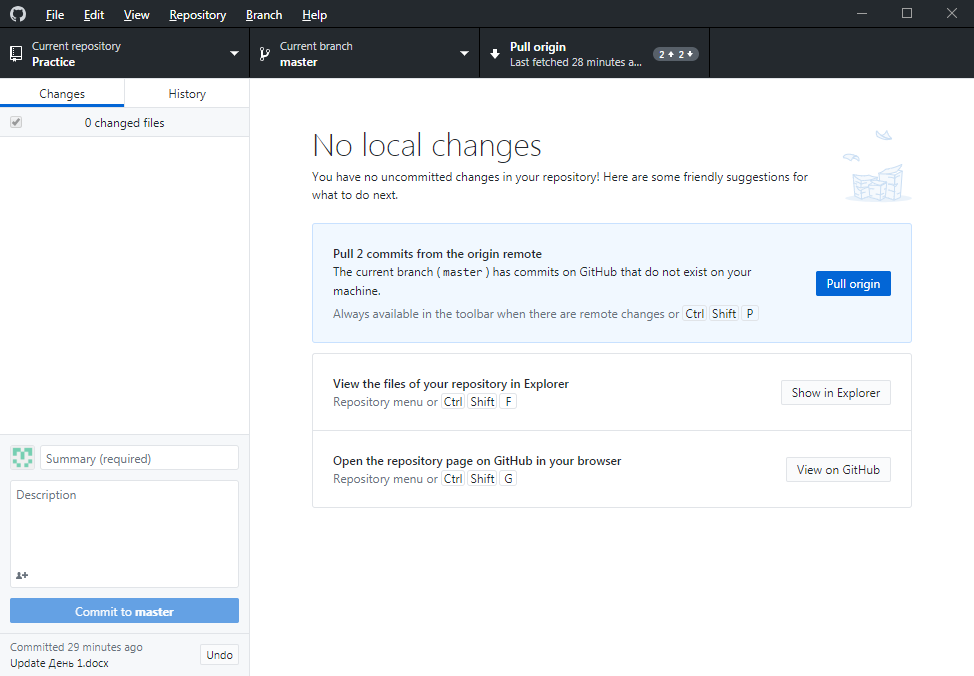
Регистрация на Github:



Клиент:



**Git. Описание и применение**

Git — это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но вы можете использовать его для любых файлов на ваш вкус). С его помощью вы можете откатиться на более старую версию вашего проекта, сравнивать, анализировать, сливать изменения и многое другое. Этот процесс называется контролем версий. Существуют различные системы для контроля версий. Вы, возможно, о них слышали: SVN, Mercurial, Perforce, CVS, Bitkeeper и другие.

Git является распределенным, то есть не зависит от одного центрального сервера, на котором хранятся файлы. Вместо этого он работает полностью локально, сохраняя данные в папках на жестком диске, которые называются репозиторием. Тем не менее, вы можете хранить копию репозитория онлайн, это сильно облегчает работу над одним проектом для нескольких людей. Для этого используются сайты вроде github и bitbucket.

**Создание нового репозитория**

Как мы отметили ранее, git хранит свои файлы и историю прямо в папке проекта. Чтобы создать новый репозиторий, нам нужно открыть терминал, зайти в папку нашего проекта и выполнить команду init. Это включит приложение в этой конкретной папке и создаст скрытую директорию .git, где будет храниться история репозитория и настройки.  
Создайте на рабочем столе папку под названием git\_exercise. Для этого в окне терминала введите:



|  |  |
| --- | --- |
| 1  2  3 | $ mkdir Desktop/git\_exercise/  $ cd Desktop/git\_exercise/  $ git init |

Командная строка должна вернуть что-то вроде:



|  |  |
| --- | --- |
| 1 | Initialized empty Git repository in /home/user/Desktop/git\_exercise/.git/ |

Это значит, что наш репозиторий был успешно создан, но пока что пуст.

**Подготовка файлов**

В git есть концепция области подготовленных файлов. Можно представить ее как холст, на который наносят изменения, которые нужны в коммите. Сперва он пустой, но затем мы добавляем на него файлы (или части файлов, или даже одиночные строчки) командой add и, наконец, коммитим все нужное в репозиторий (создаем слепок нужного нам состояния) командой commit.  
В нашем случае у нас только один файл, так что добавим его:



|  |  |
| --- | --- |
| 1 | $ git add hello.txt |

Если нам нужно добавить все, что находится в директории, мы можем использовать



|  |  |
| --- | --- |
| 1 | $ git add -A |

Проверим статус снова, на этот раз мы должны получить другой ответ:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $ git status  On branch master  Initial commit  Changes to be committed:  (use "git rm --cached ..." to unstage)  new file: hello.txt |

Файл готов к коммиту. Сообщение о состоянии также говорит нам о том, какие изменения относительно файла были проведены в области подготовки — в данном случае это новый файл, но файлы могут быть модифицированы или удалены.

**Коммит(фиксация изменений)**

Коммит представляет собой состояние репозитория в определенный момент времени. Это похоже на снапшот, к которому мы можем вернуться и увидеть состояние объектов на определенный момент времени.  
Чтобы зафиксировать изменения, нам нужно хотя бы одно изменение в области подготовки (мы только что создали его при помощи git add), после которого мы может коммитить:



|  |  |
| --- | --- |
| 1 | $ git commit -m "Initial commit." |

Эта команда создаст новый коммит со всеми изменениями из области подготовки (добавление файла hello.txt). Ключ -m и сообщение «Initial commit.» — это созданное пользователем описание всех изменений, включенных в коммит. Считается хорошей практикой делать коммиты часто и всегда писать содержательные комментарии.

## **Основные команды**

Всего несколько команд нужно для базового варианта использования Git для ведения истории изменений.

**git add**

Команда git add добавляет содержимое рабочей директории в индекс (staging area) для последующего коммита. По умолчанию git commit использует лишь этот индекс, так что вы можете использовать git add для сборки слепка вашего следующего коммита.

Это одна из ключевых команд Git, мы упоминали о ней десятки раз на страницах книги. Ниже перечислены наиболее интересные варианты использования этой команды.

Знакомство с этой командой происходит в главе [Отслеживание новых файлов](https://git-scm.com/book/ru/v2/ch00/r_tracking_files).

О том как использовать git add для разрешения конфликтов слияния написано в главе [Основные конфликты слияния](https://git-scm.com/book/ru/v2/ch00/r_basic_merge_conflicts).

В главе [Интерактивное индексирование](https://git-scm.com/book/ru/v2/ch00/r_interactive_staging) показано как использовать git add для добавления в индекс лишь отдельных частей изменённого файла.

В главе [Деревья](https://git-scm.com/book/ru/v2/ch00/r_tree_objects) показано как эта команда работает на низком уровне, чтобы вы понимали, что происходит за кулисами.

**git status**

Команда git status показывает состояния файлов в рабочей директории и индексе: какие файлы изменены, но не добавлены в индекс; какие ожидают коммита в индексе. Вдобавок к этому выводятся подсказки о том, как изменить состояние файлов.

Мы познакомили вас с этой командой в главе [Определение состояния файлов](https://git-scm.com/book/ru/v2/ch00/r_checking_status), разобрали стандартный и упрощённый формат вывода. И хотя мы использовали git status повсеместно в этой книге, практически все варианты использования покрыты в указанной главе.

**git diff**

Команда git diff используется для вычисления разницы между любыми двумя Git деревьями. Это может быть разница между вашей рабочей директорией и индексом (собственно git diff), разница между индексом и последним коммитом (git diff --staged), или между любыми двумя коммитами (git diff master branchB).

Мы познакомили вас с основами этой команды в главе [Просмотр индексированных и неиндексированных изменений](https://git-scm.com/book/ru/v2/ch00/r_git_diff_staged), где показали как посмотреть какие изменения уже добавлены в индекс, а какие — ещё нет.

О том как использовать эту команду для проверки на проблемы с пробелами с помощью аргумента --check можно почитать в главе [Правила создания коммитов](https://git-scm.com/book/ru/v2/ch00/r_commit_guidelines).

Мы показали вам как эффективно сравнивать ветки используя синтаксис git diff A...B в главе [Определение применяемых изменений](https://git-scm.com/book/ru/v2/ch00/r_what_is_introduced).

В главе [Продвинутое слияние](https://git-scm.com/book/ru/v2/ch00/r_advanced_merging) показано использование опции -w для скрытия различий в пробельных символах, а также рассказано как сравнивать конфликтующие изменения с опциями --theirs, --ours и --base.

Использование этой команды с опцией --submodule для сравнения изменений в субмодулях показано в главе [Начало работы с подмодулями](https://git-scm.com/book/ru/v2/ch00/r_starting_submodules).

**git difftool**

Команда git difftool просто запускает внешнюю утилиту сравнения для показа различий в двух деревьях, на случай если вы хотите использовать что-либо отличное от встроенного просмотрщика git diff.

Мы лишь вкратце упомянули о ней в главе [Просмотр индексированных и неиндексированных изменений](https://git-scm.com/book/ru/v2/ch00/r_git_diff_staged).

**git commit**

Команда git commit берёт все данные, добавленные в индекс с помощью git add, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.

Вы познакомились с основами модели коммитов в главе [Коммит изменений](https://git-scm.com/book/ru/v2/ch00/r_committing_changes). Там же мы продемонстрировали использование опций -a для добавления всех изменений в индекс без использования git add, что может быть удобным в повседневном использовании, и -m для передачи сообщения коммита без запуска полноценного редактора.

В главе [Операции отмены](https://git-scm.com/book/ru/v2/ch00/r_undoing) мы рассказали об опции --amend, используемой для изменения последнего совершённого коммита.

В главе [О ветвлении в двух словах](https://git-scm.com/book/ru/v2/ch00/r_git_branches_overview) мы более подробно познакомились с тем, что делает команда git commit и почему она делает это именно так.

Мы показали вам как подписывать ваши коммиты, используя опцию -S в главе [Подпись коммитов](https://git-scm.com/book/ru/v2/ch00/r_signing_commits).

И наконец мы заглянули внутрь команды git commit в главе [Commit Objects](https://git-scm.com/book/ru/v2/ch00/r_git_commit_objects) и узнали что она делает за кулисами.

**git reset**

Команда git reset, как можно догадаться из названия, используется в основном для отмены изменений. Она изменяет указатель HEAD и, опционально, состояние индекса. Также эта команда может изменить файлы в рабочей директории при использовании параметра --hard, что может привести к потере наработок при неправильном использовании, так что убедитесь в серьёзности своих намерений прежде чем использовать его.

Мы рассказали об основах использования git reset в главе [Отмена подготовки файла](https://git-scm.com/book/ru/v2/ch00/r_unstaging), где эта команда использовалась для удаления файла из индекса, добавленного туда с помощью git add.

В главе [Раскрытие тайн reset](https://git-scm.com/book/ru/v2/ch00/r_git_reset), полностью посвящённой этой команде, мы разобрались в деталях её использования.

Мы использовали git reset --hard чтобы отменить слияние в главе [Прерывание слияния](https://git-scm.com/book/ru/v2/ch00/r_abort_merge), там же было продемонстрировано использование команды git merge --abort для этих целей, которая работает как обёртка над git reset.

**git rm**

Команда git rm используется в Git для удаления файлов из индекса и рабочей директории. Она похожа на git add с тем лишь исключением, что она удаляет, а не добавляет файлы для следующего коммита.

Мы немного разобрались с этой командой в главе [Удаление файлов](https://git-scm.com/book/ru/v2/ch00/r_removing_files), показали как удалять файлы из рабочей директории и индекса и только из индекса, используя флаг --cached.

Ещё один вариант использования git rm приведён в главе [Removing Objects](https://git-scm.com/book/ru/v2/ch00/r_removing_objects), где мы вкратце объяснили как использовать опцию --ignore-unmatch при выполнении git filter-branch, которая подавляет ошибки удаления несуществующих файлов. Это может быть полезно для автоматически выполняемых скриптов.

**git mv**

Команда git mv — это всего лишь удобный способ переместить файл, а затем выполнить git addдля нового файла и git rm для старого.

**git clean**

Команда git clean используется для удаления мусора из рабочей директории. Это могут быть результаты сборки проекта или файлы конфликтов слияний.

**Kickstarter** — сайт для привлечения денежных средств на реализацию творческих, научных и производственных проектов по схеме краудфандинга (то есть добровольных пожертвований). Kickstarter финансирует разнообразные проекты, такие, как создание фильмов [независимого кинематографа США](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D1%8B%D0%B9_%D0%BA%D0%B8%D0%BD%D0%B5%D0%BC%D0%B0%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84_%D0%A1%D0%A8%D0%90), [комиксов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B8%D0%BA%D1%81%D1%8B), [видеоигр](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%B8%D0%B3%D1%80%D0%B0), музыки и т. д.

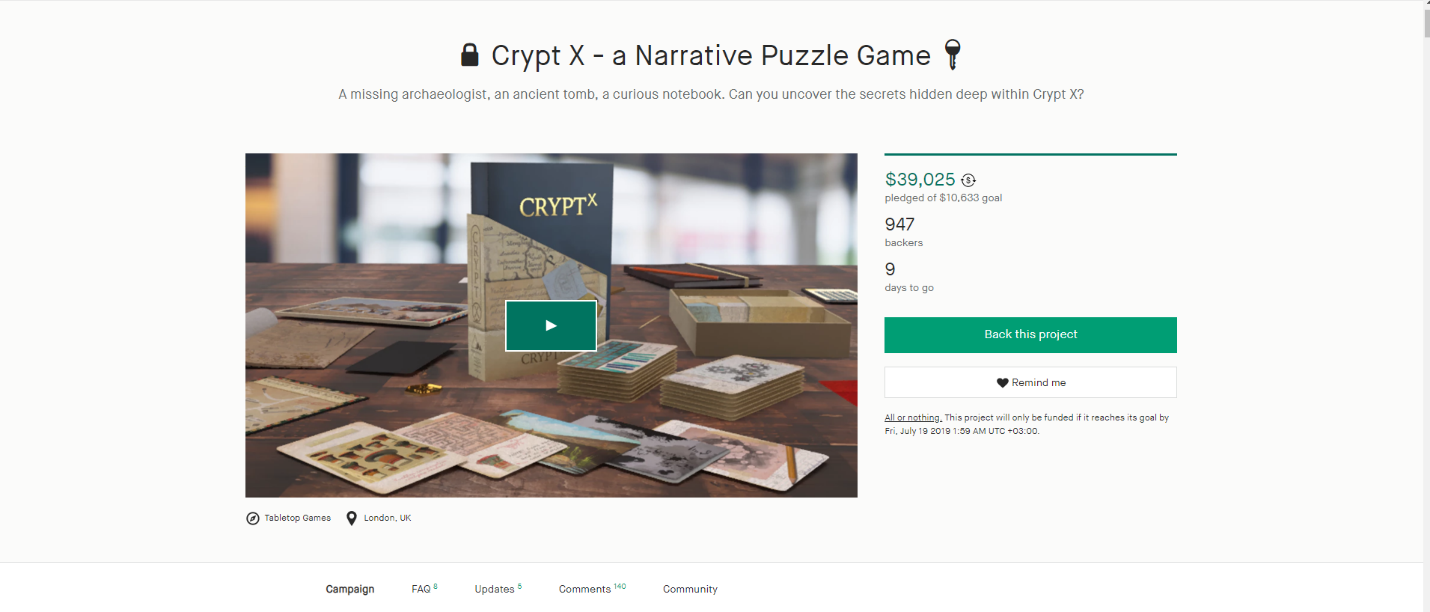


Используемый на сайте способ коллективного сбора средств называют «[краудфандинг](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D1%83%D0%B4%D1%84%D0%B0%D0%BD%D0%B4%D0%B8%D0%BD%D0%B3)».

Краудфандинг - коллективное сотрудничество людей ([доноров](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BD%D0%BE%D1%80)), которые добровольно объединяют свои деньги или другие ресурсы вместе, как правило, через [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), чтобы поддержать усилия других людей или организаций ([реципиентов](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%86%D0%B8%D0%BF%D0%B8%D0%B5%D0%BD%D1%82)). Сбор средств может служить различным целям — помощи пострадавшим от стихийных бедствий, поддержке со стороны болельщиков, поддержке политических кампаний, финансированию [стартап-компаний](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%80%D1%82%D0%B0%D0%BF) и [малого предпринимательства](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BB%D0%BE%D0%B5_%D0%BF%D1%80%D0%B5%D0%B4%D0%BF%D1%80%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE), созданию [свободного программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), получению прибыли от совместных инвестиций и многому другому.

Изначально должна быть заявлена цель, определена необходимая денежная сумма, составлена калькуляция всех расходов, а информация по ходу сбора средств должна быть открыта для всех.

Kickstarter облегчает сбор денежных средств, создав модель, которая может быть лучше традиционных способов инвестирования. Тот, кто хочет получить финансирование, должен зарегистрироваться и разместить описание проекта на Kickstarter. Kickstarter содержит рекомендации, какие проекты будут приняты.



Кикстартер работает просто. Каждый предполагаемый проект имеет свой бюджет и срок выполнения. Сумма бюджета исчисляется в долларах, а срок охватывает от 1 до 90 дней. В принципе, создатели устанавливают обе цифры. В случае достижения проектом крайнего срока, создатели смотрят на результат. Здесь возможны два сценария. Первый — это когда необходимая сумма набралась или инвестированных средств поступило больше, чем ожидалось. При втором варианте финансирование оказалось неполным, необходимые средства не набрались.

В первом варианте средства, поступившие на счет, отдаются создателю проекта, и он теперь отвечает за реализацию идеи и предоставление инвесторам наград, которые были изначально обещаны.

Если сработал второй вариант, тогда все средства, которых не хватило для набора нужной суммы, подлежат возврату на счета людей, которые эти деньги предоставили. Сервис работает по таким условиям. Если проект профинансирован успешно и нужная сумма собрана, тогда создатели Kickstarter получают 5% от общего сбора. Если требуемые деньги не собрались, сбор отсутствует.

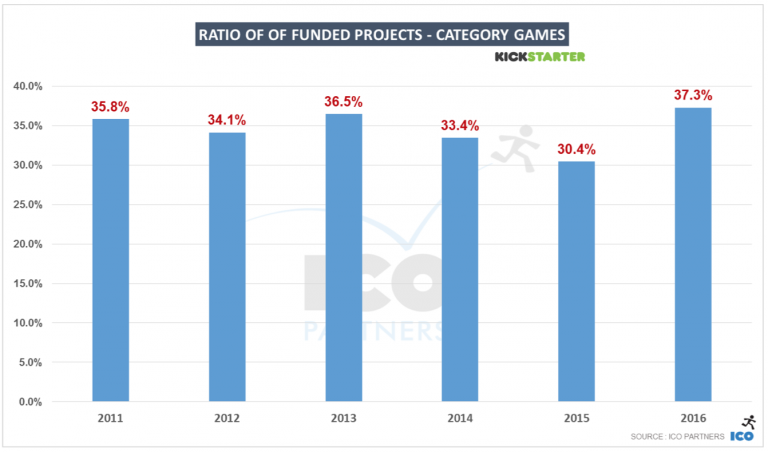
Kickstarter берёт 5 % от привлечённых средств; Stripe взимает дополнительные 3—5 %. В отличие от многих форумов по сбору средств или [инвестиций](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B2%D0%B5%D1%81%D1%82%D0%B8%D1%86%D0%B8%D0%B8), Kickstarter не претендует на право собственности на проекты и работы, которые они принимают к публикации на своем сайте, но имеют право использовать полученную от Вас интеллектуальную собственность в любых целях, не эксклюзивно. Тем не менее, проекты, осуществляемые на сайте, сохраняются и доступны для общественности. После того, как финансирование проектов завершается, загруженная информация и материалы не могут быть отредактированы или удалены с сайта.

Существует программа **Kicking It Forward**, в которой к марту 2013 приняло участие 233 проекта (в том числе 31, находившийся на стадии сбора средств): их создатели дают обещание (не контролируемое каким-либо формальным договором) 5 % прибыли, полученной после их выпуска, потратить на финансирование других проектов на Kickstarter’e.

Крупнейшие проекты по количеству привлеченных средств:

**Pebble Time** - [умные часы](https://ru.wikipedia.org/wiki/%D0%A3%D0%BC%D0%BD%D1%8B%D0%B5_%D1%87%D0%B0%D1%81%D1%8B) с цветным экраном E-Paper; обновленная версия часов [Pebble](https://ru.wikipedia.org/wiki/Pebble" \o "Pebble). Было собрано 20 338 986$

**Coolest Cooler** - многофункциональный походный холодильник. Было собрано 13 285 226$



Каждый проект должен проходить [модерацию](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80) и наблюдается тенденция, что организаторы, в основном, отбирают только потенциально наиболее успешные проекты, к которым обязательно должно быть добавлено видео, обещающее инвесторам проекта награды, бенефиты, благодарности. Например, готовый экземпляр или несколько, игры, [комикса](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B8%D0%BA%D1%81), [книги](https://ru.wikipedia.org/wiki/%D0%9A%D0%BD%D0%B8%D0%B3%D0%B0); гравировки имени инвесторов. К тому же на сайте очень трудно найти неуспешные проекты, которые не получили финансирования. Именно этим отчасти и объясняется огромный показатель успешно профинансированных проектов. По некоторым мнениям, сайт слишком сконцентрирован на успешности и слишком коммерциализирован, поэтому выступает скорее [биржей](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%80%D0%B6%D0%B0) или магазином [предоплаты](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D0%B0%D0%BD%D1%81) продуктов искусства, которые понравились.

Создание проектов разрешено только для резидентов и компаний из США, Великобритании, Канады, Австралии, Новой Зеландии, Нидерландов, Швеции, Дании, Норвегии, Ирландии, Германии, Франции, Испании, Италии, Австрии, Бельгии, Люксембурга и Швейцарии; жертвователи же могут быть со всего мира.

#### **Примеры успешных проектов с Kickstarter**

К примеру, небольшая фирма Tiktok воспользовалась услугами Kickstarter и набрала 940 тысяч долларов для производства [многофункционального зажима для iPod Nano](http://appleinsider.ru/ipod/kickstarter-syre-komu-bluetooth-v-ipod-nano.html). Спонсорами стали 13500 человек и собрали почти миллион долларов. Разработчики же выставили бюджет проекта — 15 000 долларов. Все, кто переслал 500 и больше долларов, стали владельцами специальной модели с подписью создателя, плюс 5 обычных зажимов для подарков друзьям и знакомым.

Такой же путь выбрали и дизайнеры из Нью - Йорка Том Джерард и Дэн Провост. Они посетили Kickstarter.com, предложив идею подставки-штатива для iPhone. Результат порадовал — инвестиции превысили начальную сумму в 50 раз. Чтобы данная «примочка» была выпущена, 5 тысяч человек собрали 125 тысяч долларов.

[Проект The Loog Guitars](http://loogguitars.com/) был представлял собой курсовую. В 2010 году Рафаэль Атияс корпел над диссертацией магистра в университете Нью - Йорка. Данный пример способен объяснить особенности культуры компании. Основная цель состоит не в увеличении прибыли, а в реализации только тех продуктов, которые уникальны и хорошо продуманы.

Старт проекта состоялся 30 марта, финишировал 22 апреля. Сам Рафаэль Атияс рассказывает, что смог запустить проект The Loog Guitars в марте 2011 только благодаря Kickstarter. Поставленная цель 15 тысяч долларов была превышена. Всего средств было собрано в 4 раза больше (65 618 $). Люди не просто перечислили деньги, но и хотят обеспечивать эту идею в жизнь.

Проект американской футуристки и теоретика Венессы Миемис, занимающейся исследованиями влияния коммуникационных технологий на культуру и общество, требовал финансирования для покрытия расходов при анализе возможностей самой доминирующей социальной сети мира – Facebook. В планах было создать видеоролик о «будущем facebook» с использованием методологии steep-анализа и привлечением социальных медиа, венчурных фирм и известных блогеров.

Старт проекта на Kickstarter произошел 1 марта. 6 апреля собралась сумма, превысившая ожидаемую (просили 5 тысяч долларов) на 37% (6,737$).

Плюсы платформы:

* если вы будете выбраны командой кикстартер в Project of the Day или Staff Pick, то это существенно увеличит приток людей на страницу вашего проекта, соответственно, ощутимо повысит шансы на сбор необходимой суммы
* нет необходимости думать о технической стороне вопроса сбора денег
* пользователям легче довериться и отдать деньги на кикстартере, нежели на вашем сайте, особенно если вы пока никому неизвестная маленькая команда
* мы получили несколько предложений от китайских производителей после размещения своего проекта на кикстартер — очевидно, они мониторят его на предмет возможного сотрудничества

Минусы платформы:

* необходимо быть резидентом или иметь знакомого в US/UK готового вам помочь, чтобы был счет в банке в одной из этих стран
* премодерация. Промо ролик и всю необходимую информацию вам все равно придется делать для потенциальных доноров, однако требования кикстартер ужесточаются и далеко не всякий проект пройдет премодерацию: нет готового прототипа? Путь в tech раздел вам закрыт. Ваш проект связан с благотворительностью? Путь закрыт. Также не каждый IT проект пройдет, если модератор посчитает его “недостаточно креативным”. Проект должен иметь четкую фазу завершения, то есть вы не получите деньги на проект не имеющий какого-то продукта на выходе.   
  Тем не менее, по их словам, порядка 75% проектов проходят премодерацию
* в случае успешного сбора денег вы заплатите 5% Kickstarter и еще 3-5% Amazon Payments + налоги в стране открытого счета
* это не совсем однозначно минус, но нужно иметь ввиду, что финансирование на кикстартер работает в ключе “все или ничего”, то есть если ваш проект не собрал полную сумму, то он не будет профинансирован вовсе

**Назначение разработки**

Практическая работа является комплексным проектом, охватывающим различные аспекты разработки ПО;

Данная практическая работа подразумевает сервис, на котором люди могут объединяться в команды для разработки проектов с целью получения гранта.

**Общие требования**

Сайт должен быть разработан с использованием системы Bootstrap 4 и PhpStorm. После окончания работ необходимо обеспечить возможность Заказчику самостоятельно вносить изменения (редактировать) в структуру и содержимое сайта.

**Требования к дизайну сайта**

Дизайн должен быть выдержан в строгих и мягких тонах. Использовать преимущественно сине-голубые оттенки. Дизайн сайта должен быть выполнен с использованием языка HTML и CSS, при необходимости для создания отдельных графических элементов допустимо использование технологии FLASH.

Сайт должен корректно отображаться в браузерах Microsoft Internet Explorer 6.0, 7.0; Mozilla FireFox 2.0, 3.0; Opera 9.0.

**Требования к функциональности сайта**

Необходимо обеспечить возможность предоставлять пользователям информацию на русском языке.

Сайт должен позволять пользователям:

1. осуществлять навигацию по сайту (переход между страницами);
2. выполнять вход на сайт как зарегистрированный пользователь для возможности просмотра конфиденциальной информации и/или добавления/редактирования содержимого сайта (при наличии соответствующих прав доступа).

Система управления сайтом должна позволять:

1. управлять страницами сайта (добавлять, удалять, изменять их содержимое);
2. управлять элементами меню;
3. добавлять/изменять/удалять элементы на сайте;
4. загружать на сайт графический материал (фото-видео изображения, различные файлы и т.п.).

**Требования к возможностям группы «Студент»:**

* + - Добавление, изменение и удаление информации профиля
    - Добавление, изменение и удаление интересов в профиле
    - Добавление, изменение и удаление списка научных работ
    - Добавление, изменение и удаление опыта разработки
    - Добавление, изменение и удаление идей для проектов
    - Вступление в команду
    - Выход из команды

**Требования к возможностям группы «Преподаватель»**

* + - Добавление, изменение и удаление информации профиля
    - Добавление, изменение и удаление интересов в профиле
    - Добавление, изменение и удаление списка научных работ
    - Добавление, изменение и удаление проектов
    - Добавление, изменение и удаление команд
    - Добавление, изменение и удаление списка участников команд

**Требования к возможностям группы «Администратор»**

* + - Добавление, изменение и удаление информации профилей других участников
    - Добавление, изменение и удаление интересов в профиле других участников
    - Добавление, изменение и удаление списка научных работ других участников
    - Добавление, изменение и удаление проектов
    - Добавление, изменение и удаление команд
    - Добавление, изменение и удаление списка участников команд
    - Регистрация новых участников
    - Удаление участников

Laravel — бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (англ. Model View Controller — модель-представление-контроллер). Laravel выпущен под лицензией MIT.

История

Laravel был создан Taylor Otwell как более функциональная альтернатива CodeIgniter, который не предусматривал различные дополнительные функции. Первый бета-релиз Laravel стал доступен 9 июня 2011 года, а Laravel 1 вышел в этом же месяце. Laravel 1 включает в себя встроенную поддержку для аутентификации, локализации, модели, представления, сессий, маршрутизации и других механизмов.

Laravel 2 был выпущен в сентябре 2011 года. Основные новые функции включают в себя поддержку контроллеров, которые сделали фреймворк полностью MVC-совместимым, встроенную поддержку для инверсии управления и систему шаблонов Blade.

Laravel 3 был выпущен в феврале 2012 года с набором новых функций, включая интерфейс командной строки (CLI) под именем "Artisan", встроенную поддержку нескольких систем управления базами данных, миграции баз данных в виде контроля версий, обработку событий. Выпуск Laravel 3 получил значительное увеличение числа пользователей, что повлияло на его популярность.

Laravel 4 был выпущен в мае 2013 года. Были произведены существенные изменения структуры Laravel, перенося ее макет в набор отдельных пакетов, распространяемых через Composer. Такая компоновка улучшила расширяемость Laravel 4, которая сочеталась с официальным регулярным расписанием релиза, охватывающим шесть месяцев между небольшими выпусками. Другие новые функции в выпуске Laravel 4 включают в себя выгрузка таблиц базы данных для первоначальной популяции, поддержку очередей сообщений, встроенную поддержку отправки различных типов электронной почты и поддержку "мягкого" удаления записей базы данных.

Laravel 5 был выпущен в феврале 2015 года в результате внутренних изменений, которые закончились перенумерацией релиза Laravel 4.3 в будущем. Новые возможности в выпуске Laravel 5 включают поддержку планирования периодически выполняемых задач через пакет Scheduler, слой абстракции Flysystem, который позволяет использовать удаленное хранилище так же, как и локальные файловые системы, улучшенную обработку активов пакета через Elixir и упрощенная аутентификация с внешней стороны через дополнительный пакет Socialite. Laravel 5 также представил новую внутреннюю структуру дерева каталогов для разработанных приложений.

Laravel 5.1, выпущенный в июне 2015 года. Является первым выпуском Laravel для долгосрочной поддержки (LTS) с запланированной возможностью исправления ошибок в течение трех лет и исправлений безопасности в течение трех лет.

Laravel 5.3, выпущенный 23 августа 2016 года. Новые функции в 5.3 сосредоточены на улучшении скорости разработки, добавив дополнительные усовершенствования для отдельных задач.

Laravel 5.4, выпущенный 24 января 2017 года. В этом выпуске есть много новых функций, таких как Laravel Dusk, Laravel Mix, Blade Components и Slots, Markdown Emails, автоматические фасады, улучшения маршрута.

Laravel 5.5, выпущенный 30 августа 2017 года. Возвращен пакет Whoops для отладки ошибок. Из нововведений, метод обработки запроса, новые директивы для шаблонизатора Blade, новая команда для миграции баз данных - migrate:fresh .

Возможности

Ключевые особенности, лежащие в основе архитектуры Laravel:

Пакеты (англ. packages) — позволяют создавать и подключать модули в формате Composer к приложению на Laravel. Многие дополнительные возможности уже доступны в виде таких модулей.

Eloquent ORM — реализация шаблона проектирования ActiveRecord на PHP. Позволяет строго определить отношения между объектами базы данных. Стандартный для Laravel построитель запросов Fluent поддерживается ядром Eloquent.

Логика приложения — часть разрабатываемого приложения, объявленная либо при помощи контроллеров, либо маршрутов (функций-замыканий). Синтаксис объявлений похож на синтаксис, используемый в каркасе Sinatra.

Обратная маршрутизация связывает между собой генерируемые приложением ссылки и маршруты, позволяя изменять последние с автоматическим обновлением связанных ссылок. При создании ссылок с помощью именованных маршрутов Laravel автоматически генерирует конечные URL.

REST-контроллеры — дополнительный слой для разделения логики обработки GET- и POST-запросов HTTP.

Автозагрузка классов — механизм автоматической загрузки классов PHP без необходимости подключать файлы их определений в include. Загрузка по требованию предотвращает загрузку ненужных компонентов; загружаются только те из них, которые действительно используются.

Составители представлений (англ. view composers) — блоки кода, которые выполняются при генерации представления (шаблона).

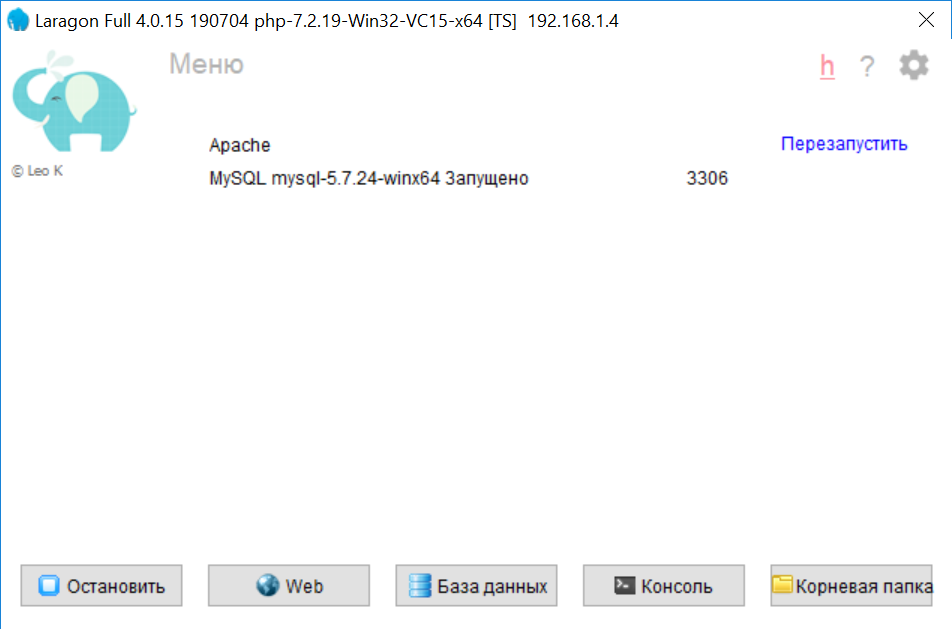
Инверсия управления (англ. Inversion of Control) — позволяет получать экземпляры объектов по принципу обратного управления. Также может использоваться для создания и получения объектов-одиночек (англ. singleton).

Миграции — система управления версиями для баз данных. Позволяет связывать изменения в коде приложения с изменениями, которые требуется внести в структуру БД, что упрощает развёртывание и обновление приложения.

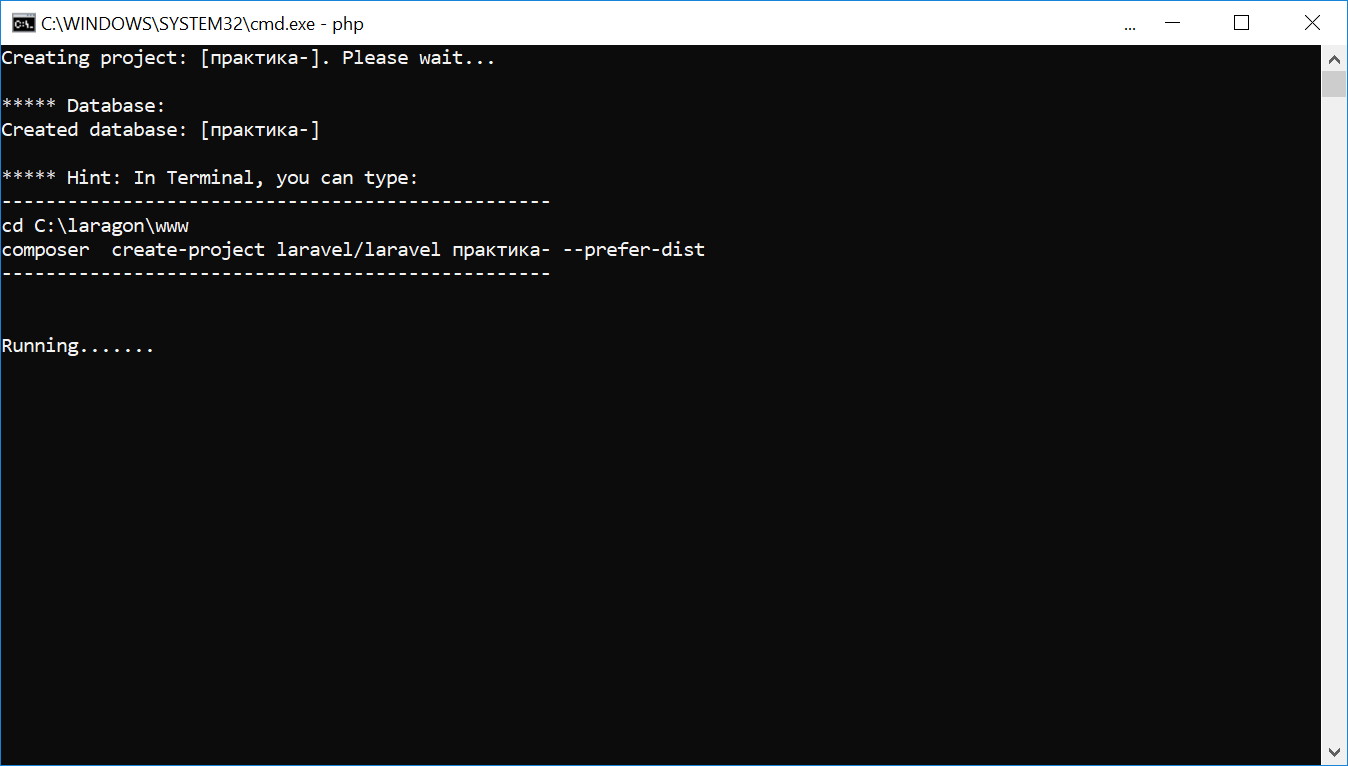
Модульное тестирование (юнит-тесты) — играет очень большую роль в Laravel, который сам по себе содержит большое число тестов для предотвращения регрессий (ошибок вследствие обновления кода или исправления других ошибок).

Страничный вывод (англ. pagination) — упрощает генерацию страниц, заменяя различные способы решения этой задачи единым механизмом, встроенным в Laravel.

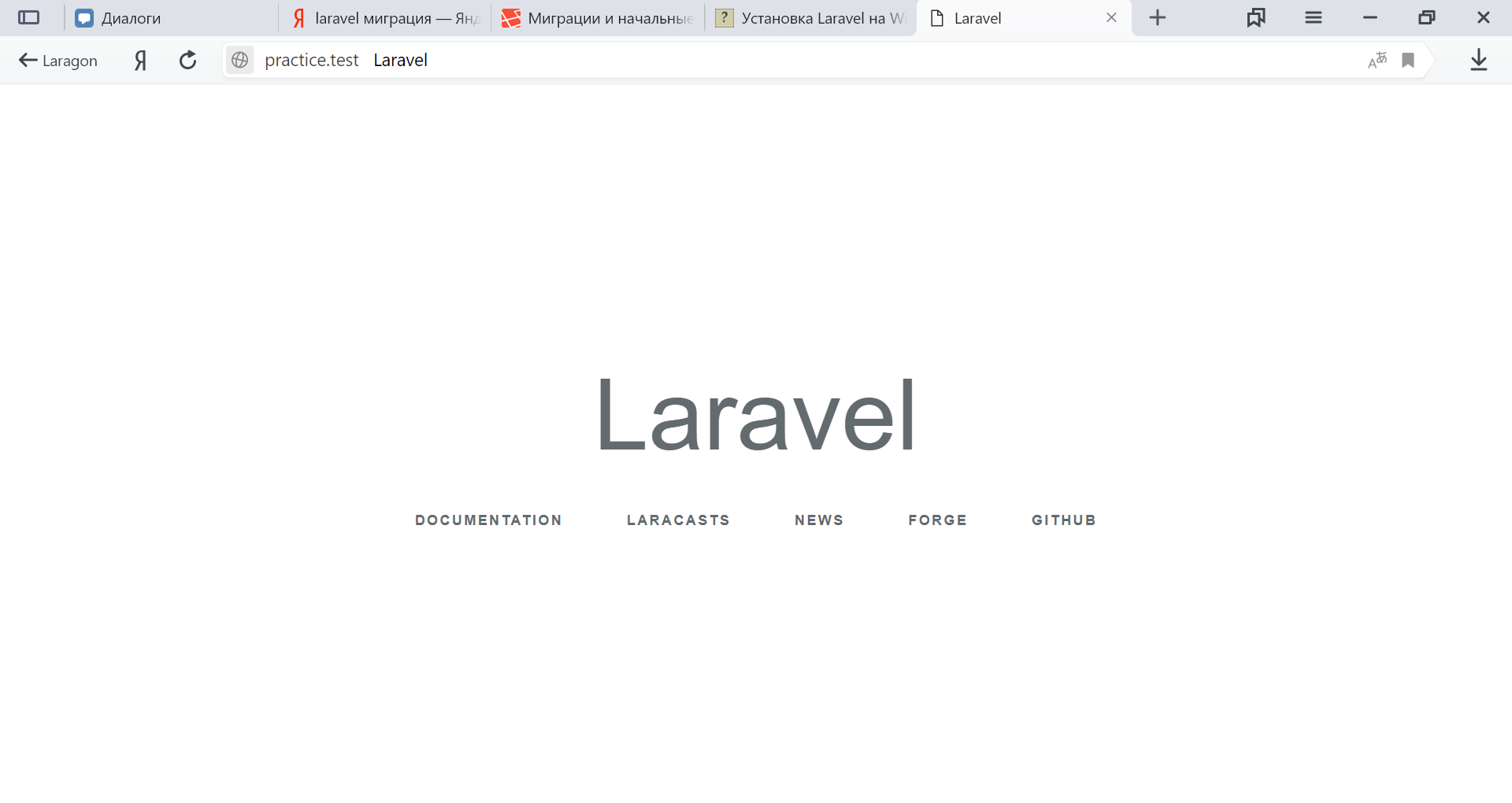
**Установка Laragon**

****

**Lavarel. Создание проекта**

****

**Лаварел. Браузер.**

****