

**ЛАБОРАТОРНАЯ РАБОТА №0.
РАБОТА В СЕМЕСТРЕ**

СОДЕРЖАНИЕ

Цель.....	2
Работа в семестре	2
Подготовительная работа	3
Алгоритм работы в семестре.....	9
Правила оформления pull request	12
Правила оформления кода.....	14
Правила сдачи лабораторных	15
Варианты	17
Отчет по лабораторным работам.....	19
Титульный лист отчета.....	20

Цель

Ознакомиться с процессом выполнения работы в семестре.

Работа в семестре

В учебном плане предусмотрено 8 лабораторных работ. Работы выстроены так, что в течении семестра идет постепенное развитие **одного** проекта. В рамках первой лабораторной работы (или до нее) **создается проект**, в который в последующих лабораторных работах **вносятся изменения**, дополнения.

Каждая лабораторная работа разбита на 2 части: базовую и усложненную. **Базовую** часть нужно делать **обязательно**. Усложнённая часть не обязательна. Пример реализации для базовой части приводится в практикуме. Информацию для решения усложненной части следует искать в иных источниках. Для проекта создается репозиторий git. Каждая часть лабораторной работы оформляется в отдельную ветку в репозитории.

За каждую лабораторную выставляются баллы, которые будут учитываться при выставлении оценки по дисциплине на экзамене. За тему можно получить до 11 баллов:

- 1 балл за посещение занятия;
- до 5 баллов за выполнение базовой лабораторной;
- до 5 баллов за выполнение усложненной лабораторной;

Принцип простой: кто больше работал в семестре (делал не только базовые лабораторные, но и усложненные лабораторные) может претендовать на более высокую оценку на экзамене.

На лабораторных будут рассмотрены следующие темы:

- 1) Библиотеки. Паттерны проектирования. IoC-контейнеры. В рамках этой лабораторной ознакомимся как создавать приложения, состоящие из нескольких проектов, как подключать

одни проекты к другим, как используется IoC-контейнеры, паттерны проектирования.

- 2) LINQ. В рамках этой лабораторной ознакомимся с технологией LINQ, как с помощью нее можно получать данные из коллекций.
- 3) Работа с БД. В рамках этой лабораторной ознакомимся с принципами взаимодействия проектов с базами данных, как подключаться к БД, получать данные из нее, сохранять данные.
- 4) Работа с офисными пакетами. В рамках этой лабораторной ознакомимся с библиотеками, позволяющими создавать документы в разных форматах данных (как правило разнообразные отчеты в приложениях).
- 5) Клиент-серверное приложение и сериализация данных. В рамках этой лабораторной ознакомимся с принципами создания клиент-серверных приложений.
- 6) Асинхронное и параллельное программирование. В рамках этой лабораторной ознакомимся с основами асинхронного программирования, как создавать асинхронные методы, как избегать коллизий параллельной работы.
- 7) Обработка строк, регулярные выражения. В рамках этой лабораторной ознакомимся как правильно манипулировать строками и как применять регулярные выражения.
- 8) Рефлексия. В рамках этой лабораторной ознакомимся с основами рефлексии, как извлекать данные из сборок, типов, как работать с атрибутами.

Подготовительная работа

Перед началом семестра, либо в начале семестра необходимо сделать следующее (**сервер git и учетка могут быть иные, нежели указаны тут, уточняйте у преподавателя в начале семестра**):

1. Создать учетную запись на сайте git, если еще нет.


2. Создать репозиторий (репозиторий один, как для базовых, так и для усложненных лабораторных). Задать репозиторию имя по следующему правилу: «**Группа Фамилия И.О. Тема по варианту**». При создании **обязательно** указать .gitignore. Указывается .gitignore для среды разработки – Visual Studio (рисунок 0.1).

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner *

 ulstulS ▾

Repository name *

Repository ✓

Great repository names are short and memorable. Need inspiration? How about [urban-guide](#)?

Description (optional)

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)


☒ Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: VisualStudio ▾

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

Create repository

Рисунок 0.1 – Создание репозитория

3. Добавить к репозиторию проверяющего. Для этого в репозитории зайти в Settings, пункт Collaborators указать имя ulstuIS и добавить (результатом данной операции будет отправлено письмо на почту пользователя ulstuIS с предоставлением доступа в репозиторий, таким образом, преподаватель узнает о новом репозитории и свяжет его с вами).
4. Через пункт Code открыть варианты скачивания репозитория и скопировать url (рисунок 0.2).

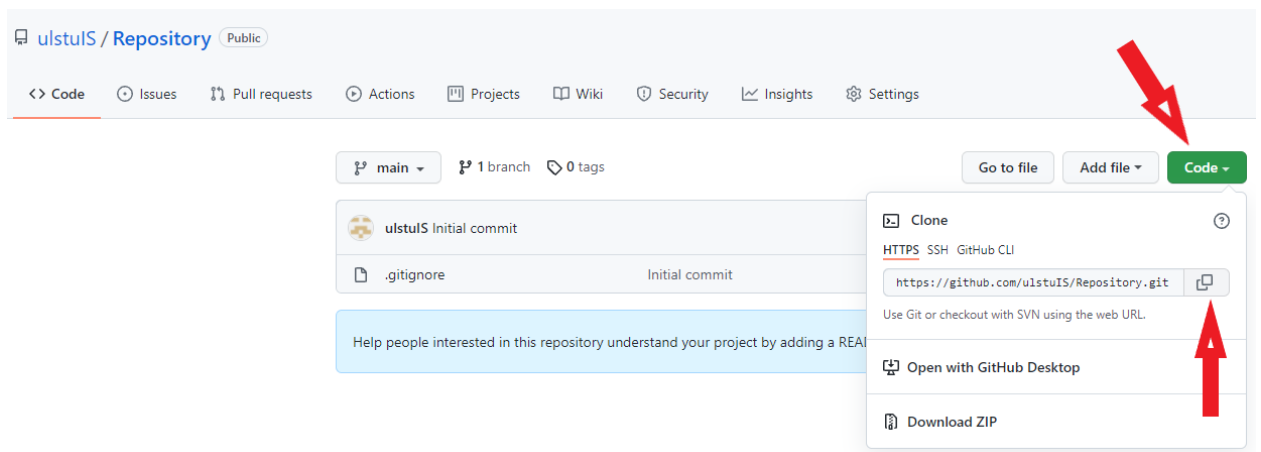


Рисунок 0.2 – Получение ссылки на репозиторий

5. Запустить Visual Studio, выбрать пункт «Клонирование репозитория» (рисунок 0.3). В новом окне вставить скопированную ранее ссылку на репозиторий и указать путь до папки назначения (рисунок 0.4).

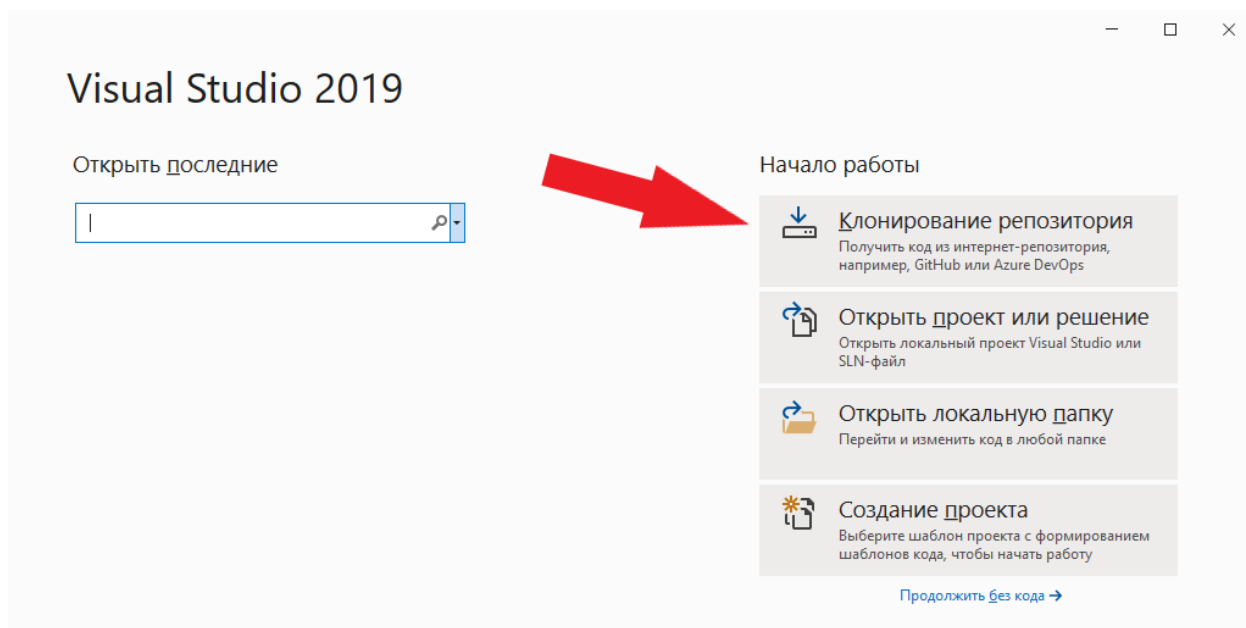


Рисунок 0.3 – Начало работ

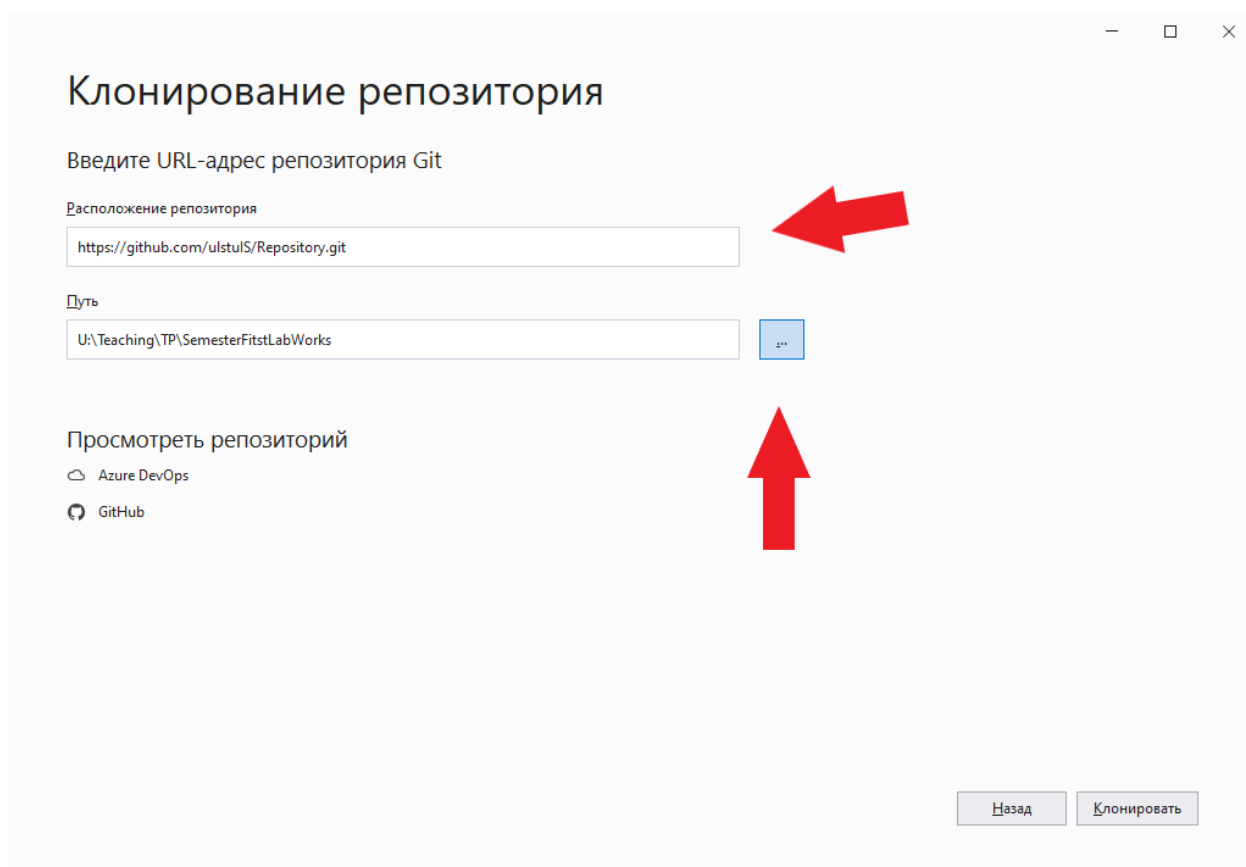


Рисунок 0.4 – Клонирование репозитория

6. Открыть вкладку «Репозиторий Git» (если не открыта, то искать в пункте меню «Вид»). По умолчанию там выбрана ветка main (master)(рисунок 0.5). Это текущая локальная ветка (клон ветки из

репозитория). А в пункте «remotes/origin» указаны ветки из глобального репозитория.

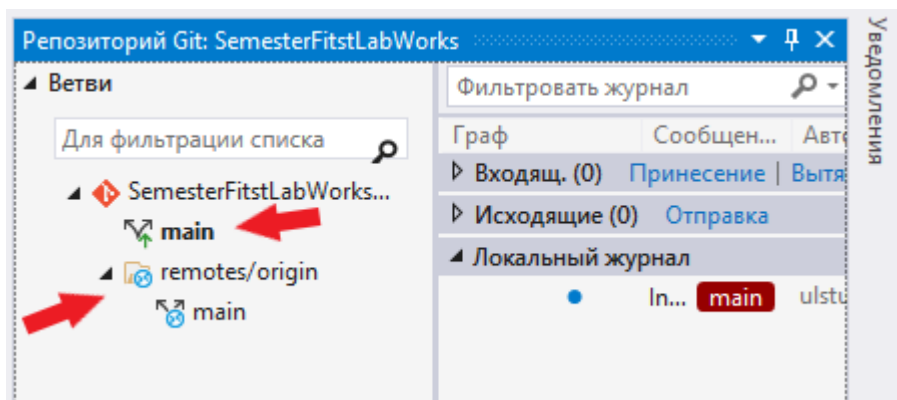


Рисунок 0.5 – Панель «Репозиторий Git»

7. Создаем проект (пункт меню «Файл -> Создать проект»).
Выбираем тип проекта – Приложение «Windows Forms»
(рисунок 0.6). Вводим название проекта (указано во варианте) и
путь (туда, куда выкачали репозиторий) (рисунок 0.7).

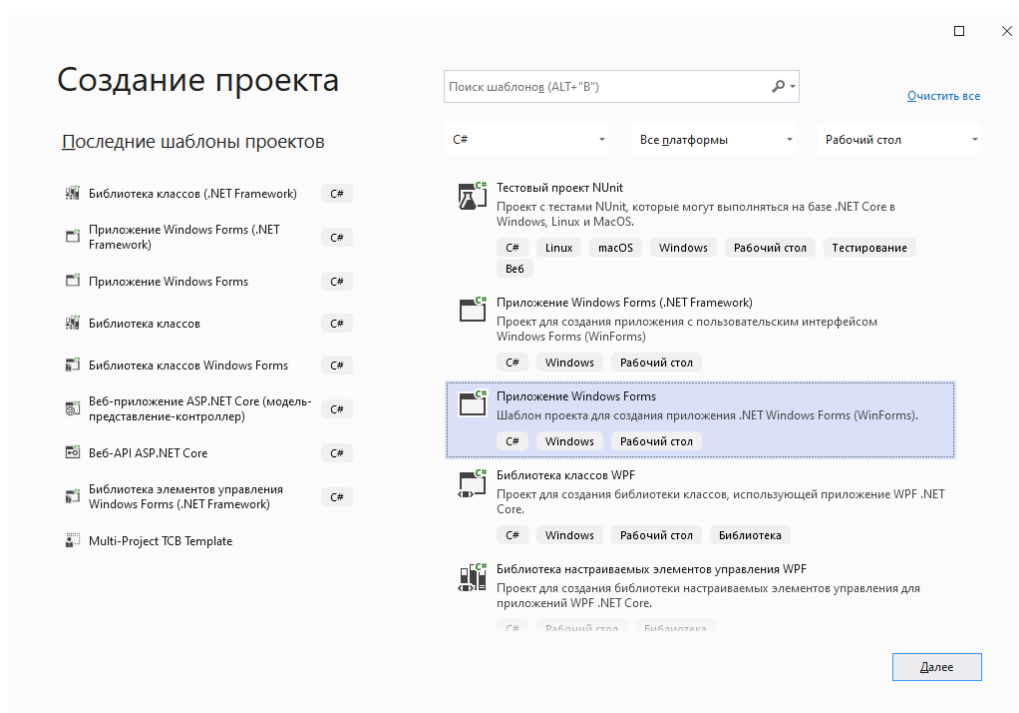


Рисунок 0.6 – Выбор типа проекта

Настроить новый проект

Приложение Windows Forms C# Windows Рабочий стол

Имя проекта
SemesterFitstLabWorks

Расположение
U:\Teaching\TP\SemesterFitstLabWorks\

Решение
Создать новое решение

Имя решения ⓘ
SemesterFitstLabWorks

☐ Поместить решение и проект в одном каталоге

Назад Далее

Рисунок 0.7 – Настройка проекта

8. Перейдем в пункт меню «Изменения git», убедимся, что в изменениях появились файлы проекта (рисунок 0.8) и отправим коммит в глобальный репозиторий (рисунок 0.9).

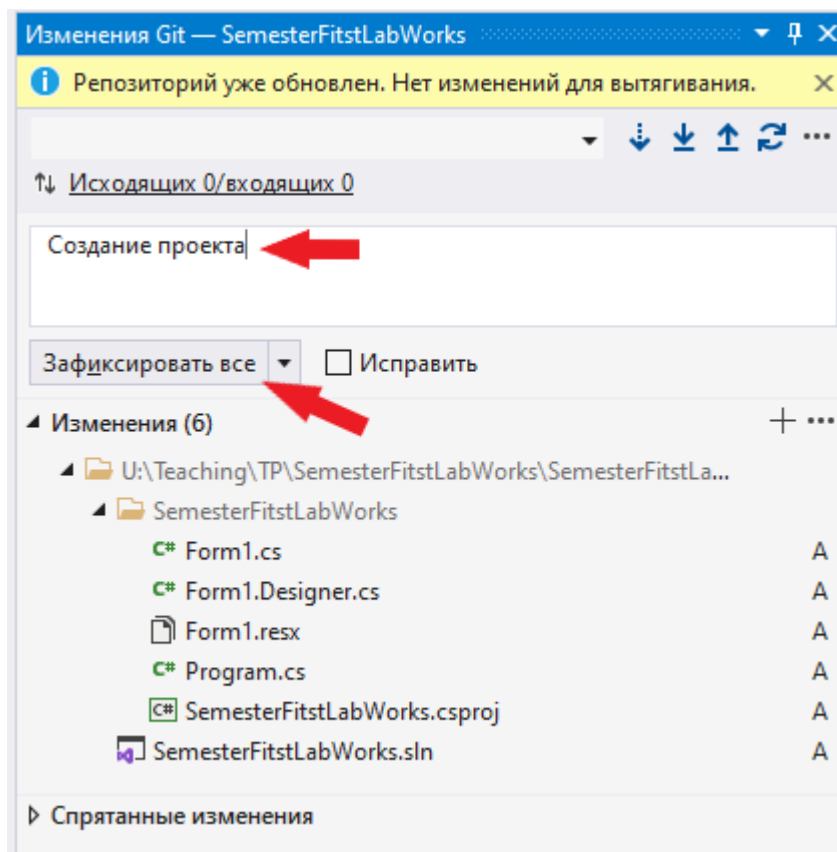


Рисунок 0.8 – Подготовка коммита

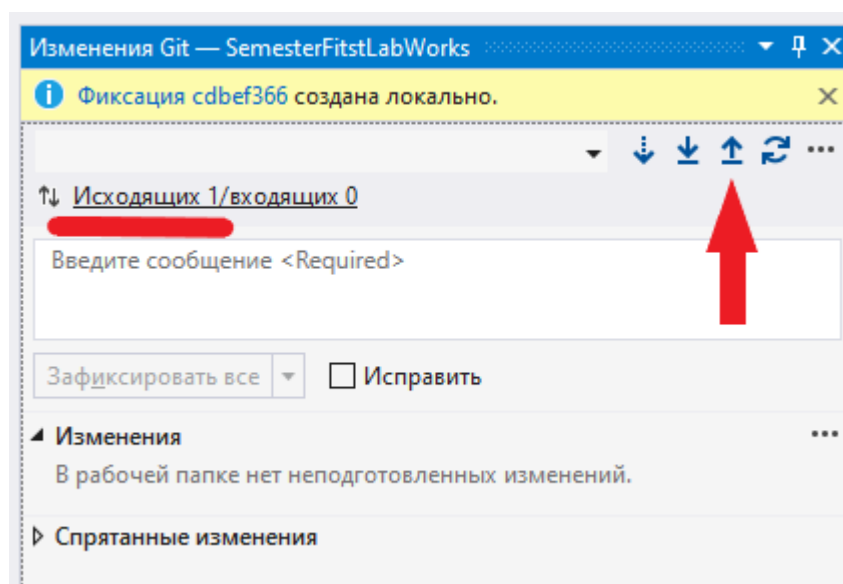


Рисунок 0.9 – Отправка коммита

Алгоритм работы в семестре

1. Ознакомится с методическими материалами по лабораторной работе. Просмотреть/прочитать лекцию по теме лабораторной работы. Просмотреть, прочитать методические указания к лабораторной работе. Для базовой части расписан и этап проектирования (по сути, как планируется писать код, какие классы/интерфейсы и т.п. могут потребоваться, для каких целей), и этап разработки (приводится код с основной логикой, оставшееся можно дописать по аналогии). Изучить код, приводимый в лабораторной работе, **понять**, как он работает, чтобы потом дописать оставшиеся фрагменты кода.
2. Создать ветку под лабораторную работу от ветки с предыдущей лабораторной работы (для первой лабораторной это будет ветка master/main). Для этого правой кнопкой кликнуть по ветке предыдущей лабораторной (master) и выбрать пункт «Создать локальную ветвь из...». Появится окно для создания новой ветке.

Вводим название ветке и в качестве исходной ветке выбираем ветку предыдущей лабораторной или master (рисунок 0.10).

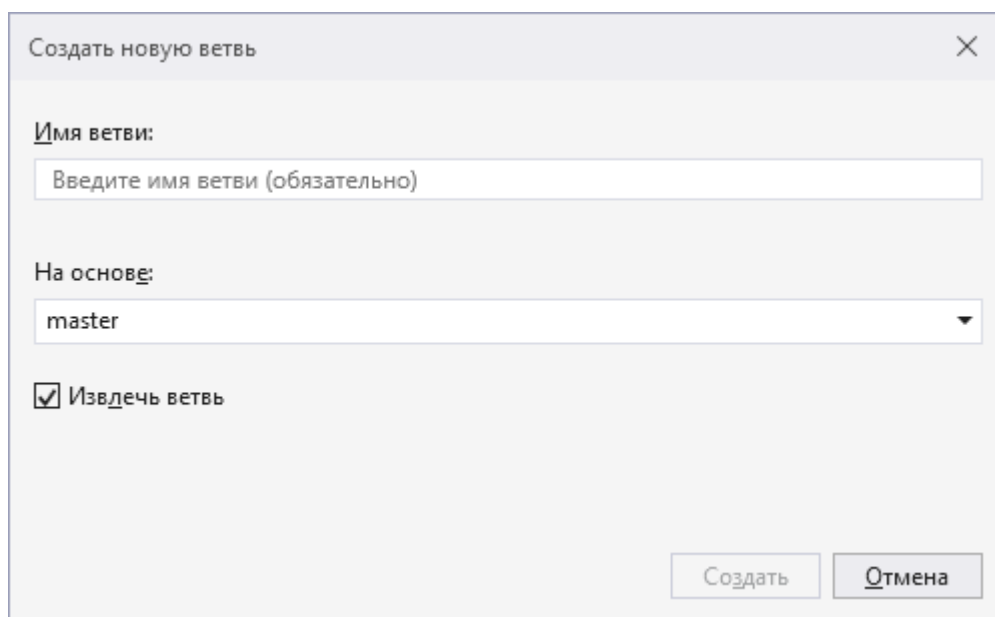


Рисунок 0.10 – Создание ветки

3. Далее сразу перейдем в пункт меню «Изменения git» и отправит коммит в глобальный репозиторий (таким образом там появится новая ветка).
4. Реализовать код, согласно заданию, к лабораторной работе с учетом всех требований.
5. Во время работы и после сдачи лабораторной создаются коммиты (их может быть неограниченное количество, также не обязательно каждый коммит сразу отправлять в глобальный репозиторий, можно накопить несколько и разом отправить). После сдачи все коммиты должны быть отправлены в удаленный репозиторий.
6. На сайте git для ветки лабораторной создать pull request (PR). Пул сравнивать с веткой от предыдущей лабораторной работы (master для первой лабораторной). **В названии пула должны обязательно фигурировать фамилия студента и номер лабораторной работы.**
7. Убедиться, что во вкладке Files присутствуют все файлы лабораторной работы, которые вы делали (классы, формы и т.п.) (рисунок 0.11). Убедиться, что код оформлен верно (см. пункт

Правила оформления кода). Если есть ошибки, исправить в коде, сделать еще ряд коммитов, отправить в глобальный репозиторий, и они добавятся в PR автоматически.

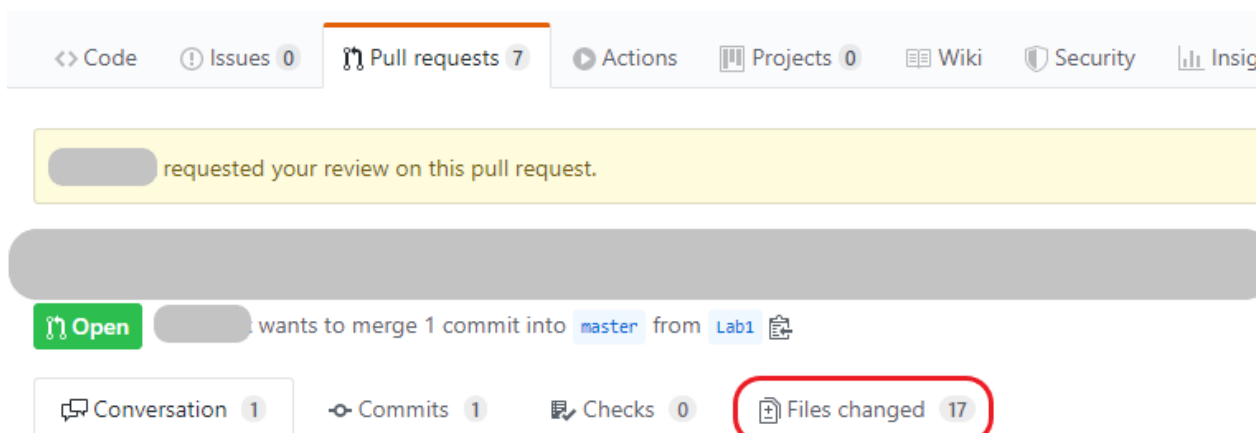


Рисунок 0.11 – Pull request

8. На вкладке Conversation справа в пункте Reviewers добавить пользователя ulstuIS (данная опция будет доступна, когда пользователь ulstuIS примет приглашение в репозиторий) (рисунок 0.12). **ВНИМАНИЕ!** Ревьюера добавлять в последнюю очередь! Когда лабораторная работа принята на занятии и проверено, что код оформлен верно.

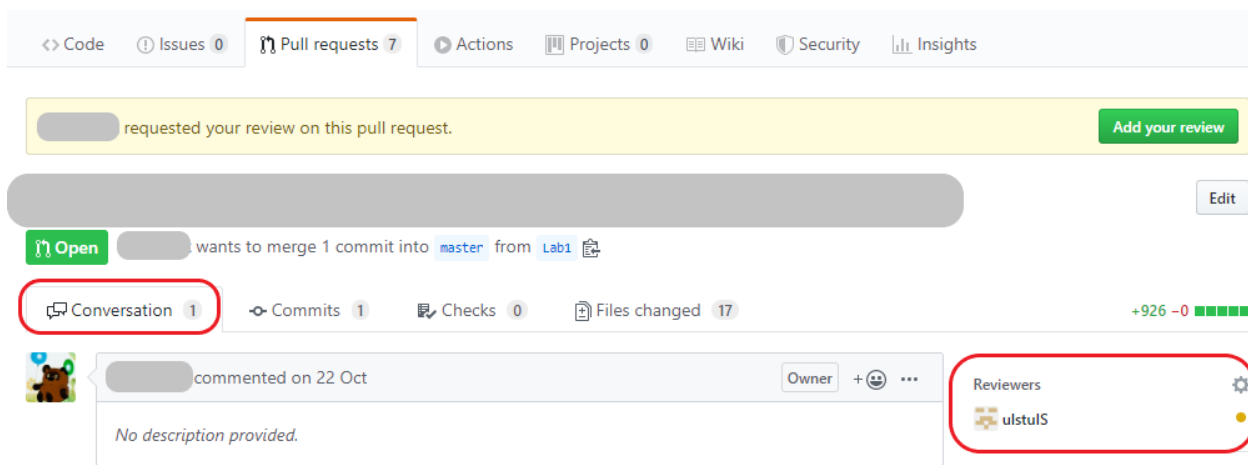


Рисунок 0.12 – Добавление проверяющего в пул

9. После проверки на вкладке Conversation могут появиться замечания по лабораторной работе (возможно, вам на почту придут письма-оповещения), либо сообщение, что лабораторная принята без замечаний.

10. Для второй базовой лабораторной следует создавать ветку от первой базовой лабораторной, для третьей от второй и т.д.
11. Pull request для **второй** лабораторной создается **не в ветку master**, а в ветку **первой** лабораторной работы. Для последующих аналогично.
12. Ветки усложненных лабораторных создаются от веток базовых лабораторных с вливанием веток с наработками по усложненной части с предыдущей лабораторной. Для первой лабораторной усложненной создается ветка от первой базовой, через коммиты заливается код, согласно заданию и создается PR к ветке первой лабораторной. Для второй лабораторной усложненной создается ветка от второй базовой и в нее вливается ветка первой усложненной (возможно, потребуется устранять конфликты слияния). PR от второй усложненной может оформляться к ветке второй базовой, либо к первой усложненной и т.д.

Правила оформления pull request

- При создании pull request убедиться, что правильно выставлены ветки откуда и куда выполняется слияние.
- Убедиться, что название пула указано верно (должны быть ФИЛ студента и номер лабораторной работы).
- В пуле отсутствуют конфликты. Конфликты возникают, если в ветку, в которую идет вливание, уже имеются изменения в тех же файлах, в которые вносились изменения в рамках задания лабораторной работы в текущей ветке. В таком случае нужно сделать или обратное слияние (т.е. отдельный pr, но ветки в нем меняются местами, и ревьюера не добавлять), любыми имеющимися средствами поправить конфликты и выполнить

слияние. Либо поправить конфликты в текущем pr средствами, предлагаемыми ресурсом.

- В пуле должны быть отражены ВСЕ изменения, которые необходимо было внести по заданию лабораторной работы (во вкладке Files посмотреть файлы).
- В пуле НЕ должны присутствовать изменения, которые НЕ требовалось делать по заданию лабораторной работы. Например, при создании pr от второй лабораторной работы на вкладке Files должно отображаться порядка 11 файлов (новые или в которые были внесены изменения). Но вы решили поменять имена переменным, забыли в рамках первой лабораторной добавить проверки или привязать действия к элементам форм и т.п. Таким образом количество измененных элементов, отображаемых во вкладке Files увеличивается. Если количество таких изменений будет превышать необходимое более чем в 2 раза, это будет считаться ошибкой. Чтобы этого избежать, нужно переключиться на ветку первой лабораторной работы, внести там изменения и затем выполнить «вливание» (pull request) наработок первой лабораторной во вторую (возможно будут конфликты, их нужно будет исправить). Таким образом у вас в pr по второй лабораторной работе останется только те изменения, которые касаются задания именно второй лабораторной работы.
- В пуле НЕ должны присутствовать файлы, которые являются компилируемыми. К ним относятся исполняемые файл, файлы библиотек и т.п. Эти файлы не являются исходными файлами проекта, зачастую зависят от платформы, на которой выполняется сборка и легко воспроизводятся на другой машине с использованием исходных файлов проекта. Такие файлы автоматически отсекаются средствами git при правильной

настройке `.gitignore` (о чем говорилось в пункте «подготовительная работа»).

- на весь семестр создается **ОДИН** проект (никаких отдельных папок, проектов), для фиксации изменений разных лабораторных использовать ветки в `git`.

В случае наличия ошибок в оформлении `pull request` будут сниматься баллы за лабораторную работу. При наличии ошибки по первым трем пунктам по 0,5 балла за каждую ошибку. При наличии ошибок за последние три пункта за `pull request` будет выставлен 0 баллов из 3-х возможных и проверка кода проводится не будет!

Правила оформления кода

- имена классов, интерфейсов, перечислений, форм и т.п. должны иметь логические имена, исходя из цели для которой они предназначены (никаких `Form1`, `Class1` и т.п.), начинаться с большой буквы, а также располагаться в файле с таким же названием.
- в файлах проекта должны присутствовать только один из базовых элементов (класс, структура, перечисление и т.п.);
- имена элементов форм, методов, свойств, полей и т.п. давать логические (исходя из их предназначения, никаких `a`, `n`, `textBox1`, `button1`), особенно для имен методов в логике форм (имена методов, связанных с элементами форм, должны иметь логические имена, связанные с этими элементами);
- заголовки форм, подписи у `label`, кнопок должны быть оформлены на одном языке и в едином стиле;
- не должно быть пустых классов;

- в классах и реализации логики форм должны отсутствовать пустые методы;
- в методах и классах недопустимо больше количество (более 3-х) пустых строк (отделяйте фрагменты комментариями или используете region для группировки кода);
- никаких закомментированных кусков кода быть не должно;
- код должен быть отформатирован.

В случае наличия ошибок в оформлении кода будут сниматься баллы за лабораторную работу (по 0,5 балла за каждую ошибку, до 3-х баллов за лабораторную, при наличии большого числа ошибок, будут и больше баллов сниматься).

Правила сдачи лабораторных

- Посещение занятий **строго обязательно!** Допускаются пропуски занятий по распоряжениям деканата (копия распоряжения), по болезни (справка из поликлиники) или по личным обстоятельствам (записка от родителей). При пропуске занятия по одной из выше описанных причин за посещение занятия будет выставлен 0 баллов (вместо 1 балла). За пропуск по неуважительной причине будет выставлен -1 балл.
- Студенты должны ходить **строго** со своей подгруппой! Студент допускается к сдаче не со своей подгруппой только по очень уважительной причине (было долгое отсутствие, до нескольких занятий, по уважительным причинам)
- К сдаче лабораторных допускаются студенты, успешно сдавшие дисциплину, связанную с программированием за предыдущий семестр (не связано с посещением, **посещение обязательно для всех!**).

- Базовая лабораторная делается и сдается на очном занятии (**по расписанию подгруппы, к которой приписан студент**).
- Допускается досдача базовой лабораторной на следующем занятии, если не успели доделать на текущем.
- По результатам сдачи выставляются 2 балла (демонстрация программы и ответа на вопрос преподавателя). В дальнейшем, при проверке pull request выставляется еще 3 балла, минус штрафные баллы при наличии ошибок при оформлении pr и кода.
- Усложненную лабораторную можно сдавать при условии сдачи всех выданных базовых лабораторных. Базовая лабораторная считается выданной, если был ее разбор на занятии. На первом занятии разбирается первая лабораторная, на втором – вторая и т.д. Таким образом, на втором занятии можно сдавать усложненные лабораторные после сдачи двух базовых. На третьем занятии – после трех базовых сданных и т.д.
- Усложненная лабораторная выполняется самостоятельно вне очного занятия. Всю необходимую для выполнения задания информацию следует искать в различных источниках. Преподаватель будет **только** принимать результат. Если у вас что-то не работает, вы сами должны найти причину и разобраться что именно не работает.
- По результатам сдачи выставляются 2 балла (демонстрация программы и ответа на вопрос преподавателя). В дальнейшем, при проверке pull request выставляется еще 3 балла, минус штрафные баллы при наличии ошибок при оформлении pr и кода.
- На занятиях (включая дополнительные) принимается не более 4-х лабораторных, из них не более 2-х усложненных лабораторных.

Приоритет (особенно на доп. занятиях) отдается базовым лабораторным (они влияют на допуск к экзамену). Т.е., если у вас не сдано 3 базовых, то вам будет разрешено сдать 3 базовых и только 1 усложненную.

- На занятии (включая дополнительные) проверяется не более 4-х pull request (из них не более 2 по усложненным) с приоритетом базовых лабораторных.
- Перед экзаменом у всех не проверенных лабораторных автоматически снимется 2 балла за непроверенный git. После экзамена пр смотреться не будут!
- **Важно!** При смене репозитория, все принятые pull request обнуляются, нужно будет выкладывать и сдавать заново (штрафные баллы проверенных PR сниматься не будут). Если вы напортачили с веткой, то достаточно сделать новую ветку от предыдущей лабораторной, либо сделать заново цепочку от ветки master, а не создавать новый репозиторий.

Варианты

Номер варианта	Название проекта	Номер варианта	Название проекта
1.	Confectionery	2.	CarRepairShop
3.	MotorPlant	4.	SushiBar
5.	ComputersShop	6.	FurnitureAssembly
7.	FishFactory	8.	SoftwareInstallation
9.	RenovationWork	10.	BlacksmithWorkshop
11.	Pizzeria	12.	PrecastConcretePlant
13.	Diner	14.	SewingDresses
15.	Typography	16.	AutomobilePlant

17.	LawFirm	18.	TravelCompany
19.	FlowerShop	20.	JewelryStore
21.	AircraftPlant	22.	GiftShop
23.	SecuritySystem	24.	FoodOrders
25.	PlumbingRepair	26.	IceCreamShop
27.	Shipyard	28.	CarpentryWorkshop
29.	Bar	30.	GarmentFactory

Отчет по лабораторным работам

Отчет включает в себя:

- Титульный лист (см приложение А).
- Описание основного приложения:
 - назначение (тут ваше творчество, исходя из варианта)
 - как работать (добавление, редактирование, удаление) с каждым из справочников (какие пол зачем нужны, со скринами);
 - путь заказа (от создания до «Оплачен»).
- Описание клиентского приложения.
 - как регистрироваться в системе и входить в систему;
 - как менять учетные данные;
 - как работать с заказами (создание, обновление, письма на почте).
- Опционально:
 - в основном приложение указать про склады (добавление, редактирование, удаление), если реализовывали;
 - при работе с заказами указать возможность нехватки материалов и как это решается (пополнение складов), если реализовывали;
 - описать складское приложение, если реализовывали.
- Листинг кода. 8 шрифт 2 колонки

Титульный лист отчета

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИСТ

Кафедра «Информационные системы»

Дисциплина «Разработка профессиональных приложений»

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ «**АБСТРАКТНЫЙ МАГАЗИН (Указать по своему варианту)**»

выполнил(а): студент(ка)
гр. ИСЭ(ПИ)бд-21
И.О. Фамилия

проверил: ст. преподаватель
Е.Н. Эгов

Ульяновск
2000 г.