1. Создание сравнительной таблицы для методологий гибкой разработки: TDD, BDD и DDD для разработки программных продуктов с достоинствами и недостатками и выводом в каких ситуациях каждую из них рекомендуется использовать. Публичная защита своей позиции.

	TDD https://docplayer.ru/52814892-Test-driven-development-razraboka-cherez-testirovanie-preimushchestva-i-nedostatki.html	BDD	DDD
смысл	процесс разработки программного обеспечения, требующий во-первых, написания автоматизированного примера для требуемой функциональности (первоначально неисправного), во-вторых, добавления минимального кода, необходимого для прохождения теста, в третьих, реорганизации кода для убеждения в том, что автоматизированные тесты до сих пор проходят.	ТЕСТ=СЦЕНАРИЙ разработка, основанная на описании поведения. ответвление подхода TDD. Нужно, во-первых, описать желаемый результат от добавляемой функциональности на предметно-ориентированном языке. Во-вторых, конструкции этого языка переводятся специалистами или специальным программным обеспечением в описание теста.	is about placing our attention at the heart of the application, focusing on the complexity that is intrinsic to the business domain itself. We also distinguish the core domain (unique to the business) from the supporting sub-domains (typically generic in nature, such as money or time), and place appropriately more of our design efforts on the core.
преиму	меньше времени на	понятные	сокращает

щества	отладку безопасно изменять или рефакторить код тесты = документация модульность полные тесты	сценарии сценарии описывают, как конечные пользователи будут использовать системы тесты = документация	издержки на сопровождение программного продукта сокращает издержки на документацию Легкая адаптация приложения к предметной области Разработчики понимают предметную область
недост атки	не всегда возможно применить ошибочный тест= ошибочный код необходимость поддержки тестов	ошибочный тест = ошибочный код необходимость поддержки тестов	Требует высокой квалификации разработчиков Много времени на анализ информации и построение модели Требует совместной работы разработчика и специалиста

2. Создание фрагмента технического задания программного продукта Web-приложение «Портфолио» с детальным описанием его ЖЦ разработки с учетом требований ГОСТ. Определение оптимальной методологии разработки этого продукта, объяснения своего выбора. Публичная защита позиции, презентация результатов работы.
https://docs.google.com/document/d/1HvpEGP3iSeTmKdn43b9HYY74e-DfTfEJFa9Uw8e6l48/edit?usp=sharing
Оптимальной методологией разработки этого ПО является гибкая методология, поскольку она подходит для проектов, которые нуждаются в срочной реализации. Также один из принципов Agile заключается в том, что “основной показатель прогресса – работающий продукт”. Это является лучшей подходящей концепцией для разработки веб сайта. В процесс

разработки можно включить применение канбан доски для визуализации целей.

3. Создание таблицы со сравнительным анализом подходов к оценки производительности (оценка в часах, story points, оценка с помощью диаграммы сгорания задач). Определение достоинств и недостатков каждого подхода. Публичная защита собственной позиции, презентация выводов.

	оценка в часах	story points	оценка с помощью диаграммы сгорания задач
преимущества	Точнее story points Руководители проектов и предприятия часто предпочитают работать с часами, т.к.последние более им знакомы. эту систему проще понять	Отсутствие взаимосвязи навыков и опыта оценщика Скорость отслеживается Нет переоценки при изменении скорости	Они ясно показывают достижения Agile team Четко показывают, что еще нужно для достижения Позволяют командам знать, находятся ли они в задаче со своими крайними сроками. Быстрое оповещение команды о потенциальных проблемах
недостатки	Оценка задачи может составлять 4 часа, но человек1 делает это через 4 часа и человек2 в 10 из-за разного уровня опыта. Вызывает проблемы пропуск тестирования, например, потому что	почти всегда неточна Чтобы определить реальную скорость, требуется некоторое время	ограничивают - диаграммы показывают только часть общей картины не показывают, какие задачи все еще выполняются не показывают, насколько близко

	у вас закончилось много часов. Медленнее сделать этот тип оценки, потому что вам нужно подробно рассчитать, чтобы определить часы. Задача может увеличиться, чтобы заполнить часы.	Легок для непонимания и неправильного использования	команда завершает свою работу. могут привести к преувеличенным ожиданиям

4. Разработка прототипа приложения «Генератор портфолио» на основе фрагмента технического задания с обоснованием целесообразности выбора технологий и архитектуры продукта (Python, Javascript, Pascal)
Для разработки веб-приложения “генератор портфолио” был выбран следующий стек технологий на клиентской стороне: React.js
Для возможного дальнейшего расширения приложения используется фреймворк, поскольку он имеет ряд преимуществ перед другими:

1. позволяет переиспользовать написанные компоненты,
2. при изменении данных(состояния) мгновенно меняется отображение
3. избавляет от необходимости манипуляций с DOM при изменении данных, вместо этого используется Virtual DOM, а потом, если в этом есть надобность, используется DOM,
4. (имеет больший рейтинг “Рассматривая данные за последние 2 года, мы можем видеть, что в противовес данным по рейтингам звезд на github React по-прежнему доминирует с точки зрения фактического использования, измеряемого загрузкой пакетов NPM”)
5. упрощение манипуляции DOM объеками

1. Создание сравнительной таблицы с описанием процесса разработки приложения «Онлайн-конвертер валют» для двух подходов к разработке: через поведение, через тестирование.

Задача	Через тестирование	Через поведение
Авторизация	*Описание постановки задачи:* Пользователь авторизуется с помощью соц. сети вконтакте. Пишем	*Описание постановки задачи:* Пользователь авторизуется с помощью соц. сети вконтакте. *Описание сценария с*

	функцию валидации введенных данных и функцию проверк ответа на запрос авторизации. Далее пишем компонент.	*помощью ключевых слов*: Пользователь нажимает на кнопку, ТОГДА появляется окно ввода логина и пароля. ПОСЛЕ ввода логина и пароля происходит авторизация ИЛИ вывод сообщения об ошибке. ПОСЛЕ авторизации мы видим имя и фамилию пользователя. Написание тестов: разделяем сценарий по ключевым словам на предложения которые являются заголовками функций теста. После написания теста, пишем компонент.
Окно редактирования портфолио	*Описание постановки задачи:* Разработать модальное окно для редактирования данных в портфолио. Пишем тесты для проверки открытия модального окна, получения информации с github по средством http запроса. После написания теста, пишем компонент.	*Описание постановки задачи:* Разработать модальное окно для редактирования данных в портфолио. Сценарий: Пользователь нажимает на кнопку открытия модального окна ПОСЛЕ, вводит ссылку на github ПОСЛЕ получает данные о профиле, ПОСЛЕ нажимает кнопку сохранить, ПОСЛЕ видит измененные данные на странице портфолио. Написание тестов: разделяем сценарий по ключевым словам на предложения которые являются заголовками функций теста. После написания теста, пишем компонент.
Окно	*Описание постановки*	*Описание постановки задачи:*

отображения портфолио	*задачи:* Разработать авто генерируемую страницу страницу портфолио исходя из входных данных. Сценарий: пользователь видит страницу. Написание тестов: Поиск и проверка в результирующем HTML документе входных данных.	Разработать авто генерируемую страницу страницу портфолио исходя из входных данных. Сценарий: пользователь видит страницу. Написание тестов: Поиск и проверка в результирующем HTML документе входных данных.

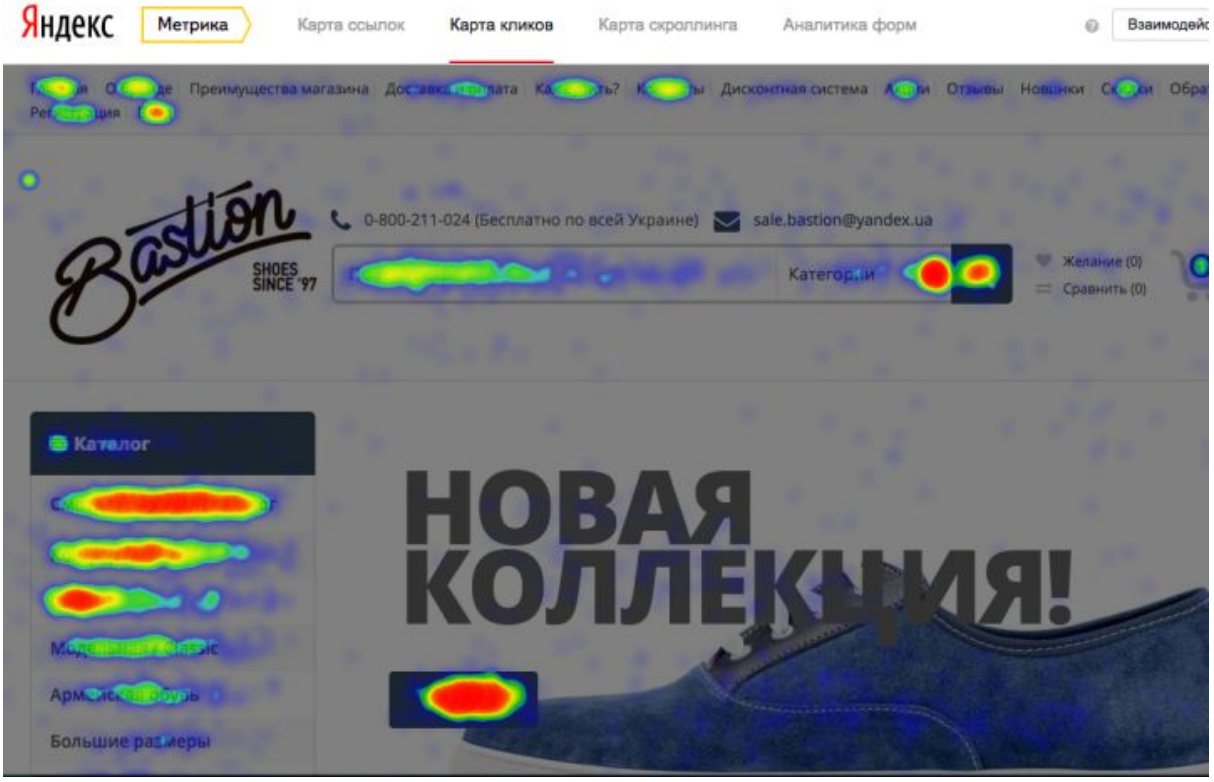
2. Разработка каркаса веб-приложения «Генератор портфолио» с использованием подхода TDD (разработка через тестирование) с фиксацией процесса разработки в системе управления проектами Trello
Первым делом, напишим тест на входные/выходные данные (функцию валидации ответа после авторизации) для первого компонента - формы для логирования пользователя через социальную сеть Вконтакте.
Затем напишим компонент авторизации.
Проверим его на прохождение теста.
Второй этап заключается в написании теста для компонента генерации портфолио (правильно ли все отображается), написании компонента и проверки.
В конце делаем рефакторинг, если тесты провалены.
доска на трелло

1. Создание презентации с обзором и анализом удачных и неудачных кейсов проектирования пользовательских интерфейсов (UI) и пользовательского опыта (UX)
https://docs.google.com/presentation/d/1BUxM4QMrzaocuEdXWjbwyOZKnZrKsdTLUGAxZewpNF0/edit?usp=sharing

Пользовательский интерфейс – это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы программной системы.

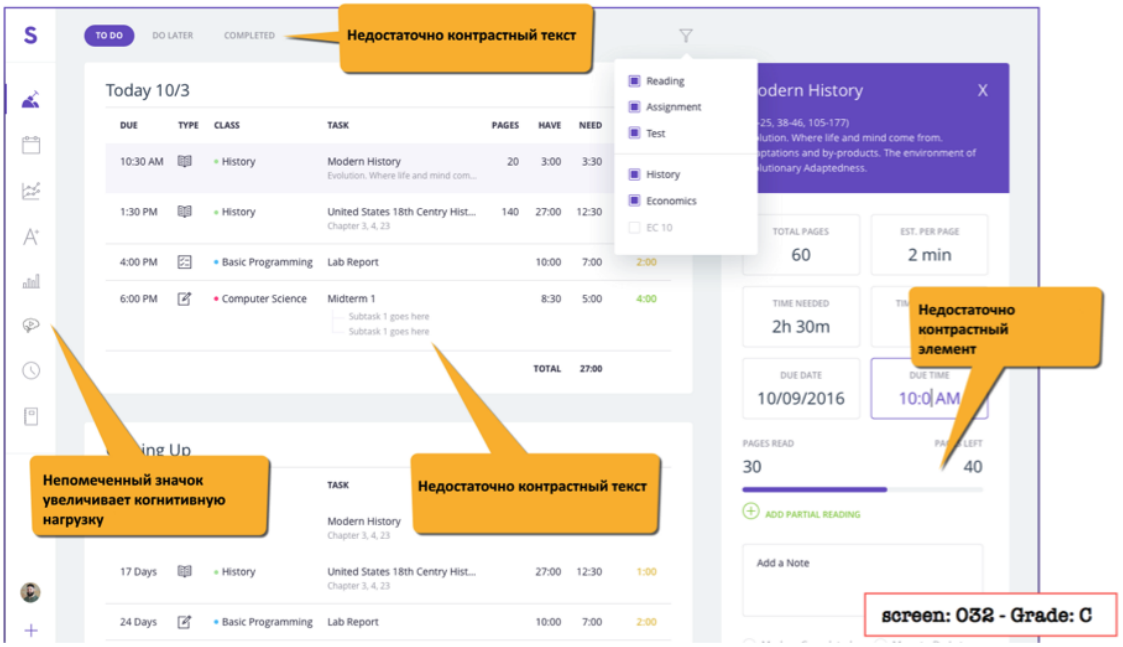
Пользовательский опыт – это совокупность эмоций, действий и результатов, полученных пользователем во время взаимодействия с системой/продуктом/сайтом.
Например, если у нас есть просто красивый интерфейс, то он не будет правильно спроектированным, т.к. он должен быть создан по определенным правилам (“Эстетичность – это лишь внешний вид, а UX-дизайн – это и внешний вид, и общие ощущения, и качество работы.”)

Для оценки пользовательского интерфейса и опыта будем использовать карту кликов Яндекс.Метрика, а для измерения целей (экспорт созданного портфолио в соц. сети и другие сервисы) будем использовать счетчик Яндекса. На основании данной статистики сделаем вывод, какое из решений наиболее подходящее.
Вот так приблизительно она будет выглядеть:



Представим два случая - модальное окно редактирования и отдельная вкладка для редактирования портфолио. Чем больше кол-ва нажатия на кнопку, тем юзабильнее интерфейс.

кейс ужасного UX



1. Создание фрагмента технического задания с использованием ООППС (инкапсуляция, модульность, наследование) для приложения «Онлайн конвертер величин». Описание классов, объектов и их поведения. Согласно стандарту IEEE STD 830-1998, пункт 3 “Детальные требования” может содержать такую структуру:

- 1. Требования к внешним интерфейсам
    - 1. Интерфейсы пользователя
    - 2. Интерфейсы аппаратного обеспечения
    - 3. Интерфейсы программного обеспечения
    - 4. Интерфейсы взаимодействия
- 2. Функциональные требования
- 3. Требования к производительности
- 4. Проектные ограничения (и ссылки на стандарты)

- 5. Нефункциональные требования (надежность, доступность, безопасность и пр.)

- 6. Другие требования

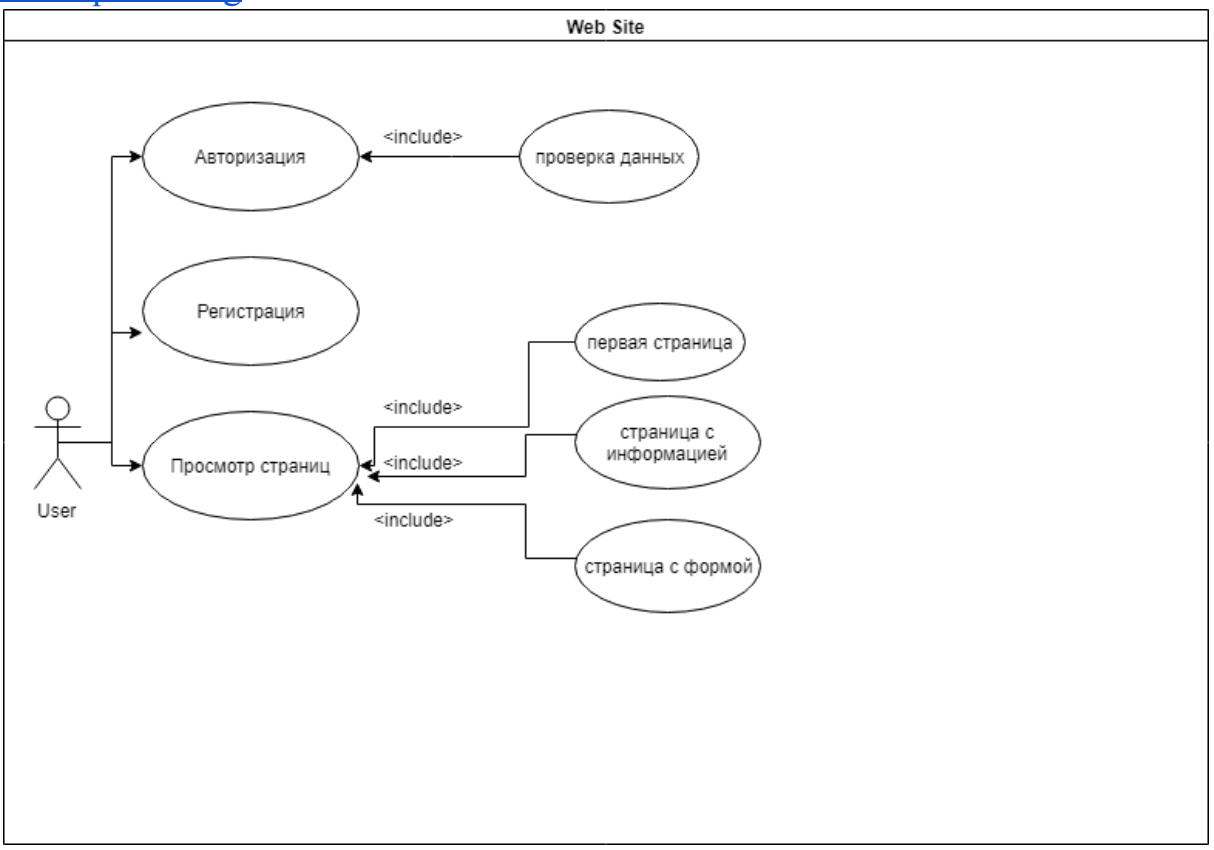
В данном задании мы будем описывать подпункт 6 “Другие требования”, в котором пропишим классы, объекты и их поведение для разрабатываемого приложения в рамках дипломного проекта - “Генератор портфолио”.
В процессе проектирования были созданы компоненты, наследующие свойства класса React.Component.
“Компонент, по сути, можно сравнить с классом. Он отличается от последнего только более мелкими размерами и более специфичными характеристиками.” (“ C++ для профессионалов”, Николас А. Солтер, Скотт Дж. Клепер)
Компонент User, Portfolio и главный - App.
Компонент User отвечает за отображение данных о пользователе, возможность их редактировать. Содержит свойство (метод) gitFetchUser, которое позволяет выгружать данные с помощью GitHub API о пользователе с введенным никнеймом.
Компонент Portfolio отображает портфолио, если есть введенные данные, если же нет их, то отображается предложение зарегистрироваться. Содержит свойство renderPortfolio, которое принимает на ввод данные о пользователе, введенные ранее, и отображает строку с введенными данными.
Компонент App является главным поскольку отображает все данные и содержит в себе роутинг двух компонентов. Также в нем реализован метоюд авторизации в vk - VkOnAuth

3. Проектирование и разработка фрагмента (сущности «Пользователь», «Заявка») приложения для регистрации на конференцию. Описание отношений между объектами.
Все компоненты, описанные выше (User, Portfolio и App), **наследуют** класс React.Component.
Компонент App содержит **композицию** двух остальных(User, Portfolio) компонентов и **реализацию** Portfolio с методом VkOnAuth.
Компонент Portfolio **ассоциирован** с User.

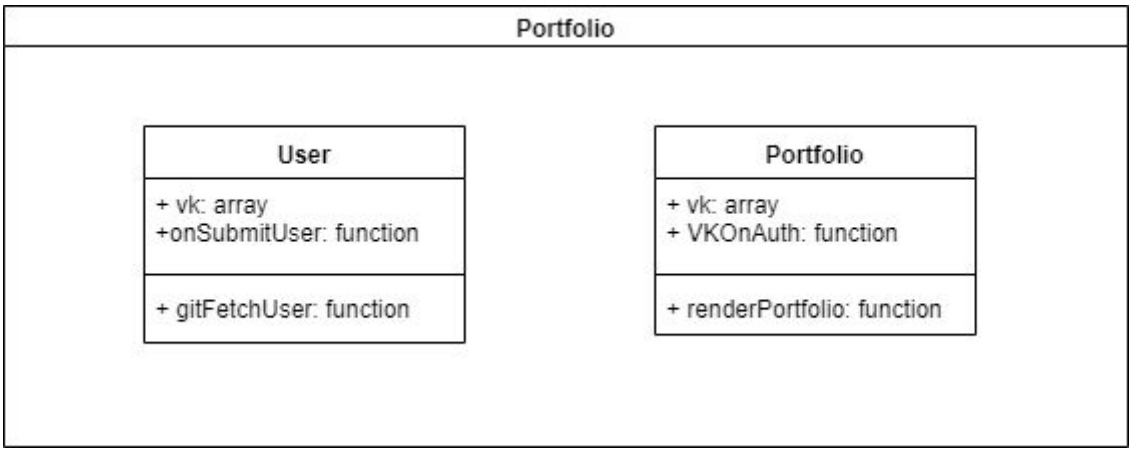
4. Разработка каркаса приложения «Онлайн-конвертер величин» с использованием схемы разделения данных MVC.

В случае проектирования по MVC у нас появляются некоторые различия с ООППС.
Первое - **представление**. Оно реализовано в компоненте Portfolio.
Второе - **контроллер.** Он заключен в компоненте App.
Третье - компонент User содержит **модель***.
*перенести все методы в этот компонент, кроме метода renderPortfolio.

3. Проектирование диаграммы прецедентов (вариантов использования) .
https://drive.google.com/file/d/1fJ6ek-S9eEWMu0F5DuZRAAMG5tYERvci/view?usp=sharing



4. Проектирование диаграммы классов и генерация на основе этой диаграммы программного кода для приложения «Гостевая книга».



ВАРИАТИВНАЯ РАБОТА:
1. Создание прототипа технического задания программного продукта с детальным описанием его ЖЦ. Определить наиболее оптимальную методологию для разработки этого продукта и объяснить свой выбор.
https://docs.google.com/document/d/1HvpEGP3iSeTmKdn43b9HYY74e-DfTfEJFa9Uw8e6l48/edit?usp=sharing
Оптимальной методологией разработки этого ПО является гибкая методология, поскольку она подходит для проектов, которые нуждаются в срочной реализации. Также один из принципов Agile заключается в том, что “основной показатель прогресса – работающий продукт”. Это является лучшей подходящей концепцией для разработки веб сайта. В процесс разработки можно включить применение канбан доски для визуализации целей.

2. Разработка электронного образовательного ресурса на тему «Использование методологии SCRUM при разработке корпоративного сайта» (составление глоссария, описания последовательности выполнения этапов, ролей участников и их действий).

https://dar-pa.moodlecloud.com/course/view.php?id=4
Логин: student
Пароль: 123456789