51. Основы тестирования

Цель:

Познакомиться со следующими понятиями процесса тестирования:

- виды тестирования
- инструменты для тестирования JS модулей (Jest, Mocha, Chai)
- тестирование React компонентов

План занятия:

- виды тестирования
    - модульное
    - интеграционное
    - функциональное
    - сквозное
    - smoke
    - приемочное
    - тестирование производительности
- инструменты для тестирования
- тестирование react-компонентов

Конспект:

Виды тестирования:

Модульные тесты

Это тип тестирования ПО, при котором тестируются отдельные модули или компоненты ПО. Цель такого тестирования заключается в том, чтобы проверить, что каждая единица кода работает должным образом.

Интеграционное тестирование

В ходе интеграционного тестирования проверяется, хорошо ли работают вместе различные модули и сервисы, используемые приложением. Например, можно убедиться, что микросервисы работают вместе так, как задумано. Этот вид тестирования является более затратным, поскольку для проведения тестов требуется запуск различных компонентов/сервисов приложения.

Функциональные тесты

В функциональных тестах основное внимание уделяется бизнес-требованиям к приложению. Они проверяют только результат некоторого действия и не проверяют промежуточные состояния системы при выполнении этого действия.

Интеграционные vs функциональные тесты

Разница интеграционного и функционального тестирования в том, что интеграционный тест нужен в программе просто чтобы убедиться, что вы можете отправлять запросы к базе данных, тогда как функциональный тест будет ожидать получения из базы данных определенного значения в соответствии с требованиями продукта.

Сквозные тесты (E2E)

Сквозное тестирование копирует поведение пользователя при работе с ПО в контексте всего приложения. Оно обеспечивает контроль того, что различные схемы действий пользователя работают должным образом. Сценарии могут быть как очень простыми (загрузка веб-страницы или вход в систему), так и гораздо более сложными (проверка почтовых уведомлений, онлайн-платежей и т. д.).

Приемочное тестирование

Приемочные тесты — это формальные тесты, которые призваны проверить, отвечает ли система требованиям бизнеса. При этом необходим запуск всего приложения, и основное внимание уделяется воспроизведению поведения пользователей. В ходе этого тестирования возможен даже замер производительности системы, и в случае несоответствия установленным требованиям внесенные изменения могут быть отклонены.

Тестирование производительности

Тесты производительности проверяют поведение системы при нахождении под значительной нагрузкой. Такие тесты служат для оценки надежности, стабильности и доступности платформы.

Smoke-тестирование

Smoke-тесты представляют собой базовые тесты, которые проверяют основные функциональные возможности приложения. Такие тесты должны выполняться быстро, поскольку их цель — убедиться, что основные возможности системы работают как запланировано.

TDD/BDD:

Что такое TDD и BDD на пальцах, и что должен знать о них фронтендер

Инструменты для тестирования:

Jest — это фреймворк для тестирования, разработанный Facebook.

Среди особенностей Jest можно отметить следующие:

- Производительность. В первую очередь стоит сказать о том, что фреймворк Jest признан самым быстрым в применении к большим проектам со множеством файлов тестов
- Пользовательский интерфейс и документация. Интерфейс Jest отличается понятностью и удобством.
- Наличие всего необходимого для начала работы. Jest поставляется со средствами создания утверждений, функций-шпионов и имитаций, которые эквивалентны отдельным библиотекам, вроде Sinon, выполняющим те же функции. Если стандартные возможности вас не устраивают и нужно что-то особенное, можно воспользоваться и сторонними библиотеками.
- Глобальные переменные. Jest создает глобальные переменные тестирования, поэтому их не нужно явно подключать

Sinon.JS представляет собой мощную автономную систему, предоставляющую возможность выполнять тестирование JavaScript-проектов с использованием так называемых шпионов (spy), заглушек (stub) и имитаций (mock). Эта система умеет работать с любыми фреймворками для модульного тестирования.

Моки и стабы: https://habr.com/ru/post/134836/

Chai – это самая популярная библиотека для создания утверждений. Библиотека поддерживает разнообразные функции для проверок.

Mocha в настоящий момент является самой широко используемой библиотекой. Эта библиотека применяет сторонние инструменты для построения утверждений и внешние средства для создания имитаций и функций-шпионов (обычно Enzyme и Chai). Это означает некоторые дополнительные сложности при настройке Mocha, и то, что получаемый функционал будет разделён между различными библиотеками. При этом мы получаем более гибкий и расширяемый фреймворк.

Вот некоторые особенности Mocha, на которые стоит обратить внимание:

- Сообщество. Разработано множество плагинов и расширений для использования в уникальных сценариях тестирования
- Расширяемость. Плагины, расширения и библиотеки, которые можно использовать с Mocha, такие, как Sinon
- Глобальные переменные. Mocha, по умолчанию, создает структуры теста в глобальном виде. Это не относится к утверждениям, имитациям, функциям-шпионам

Тестирование React компонентов

https://ru.reactjs.org/docs/testing.html