포트폴리오 작성을 위한 리액트 프로젝트 고도화 2회차

질의응답 사항중에...

1. Supabase
   a. 괜찮은 블로그를 찾아서 공유드립니다!
   b. https://jgjgill-blog.netlify.app/post/quickly-start-a-project-with-supabase/
2. HTTP protocol upgrade
   a. 서버 설정만 해주면됨
   b. Nginx, apache 버전 확인 필수

Agenda

1. 무엇을 테스트 할 것인가?
   a. 어디까지 테스트 할 것인가?
2. 어떻게 테스트 할 것인가?
   a. 유닛테스트
   b. 통합테스트
   c. E2E 테스트
3. 프론트엔드 TDD

무엇을 테스트 할 것인가?

1. UI 테스트는 진행하지 않음
    a. 로그인 화면에서 비밀번호 input과 버튼 사이의 거리를 재는 것은 무의미
        i. 기기의 크기에 따라 달라질 것
    b. 컴포넌트 렌더링 순서도 무의미
        i. JSX는 선언한대로 렌더링 된다
        ii. 인풋이 버튼 위에 존재하는지는 테스트할 필요가 없는 것
2. 요구사항의 “사용자 시나리오"에 집중해야 한다
    a. 아이디 비밀번호를 입력했을 때 버튼이 활성화 되는가?
    b. 하지만 위보다 중요한것은
        i. 로그인이 성공하는가?
        ii. 로그인이 실패하는가?
    c. “기능”에 중점을 두고 테스트를 해야한다

무엇을 테스트 할 것인가?

1. 로그인 화면에서 버튼 테스트
2. 테스트 시나리오?
    a. 버튼이 잘 보이는가
    b. 버튼이 요구조건에 맞게 활성화 되는가
    c. 활성화된 버튼을 클릭하면 이벤트가 잘 발생하는가
3. 테스트 케이스
    a. 제대로 된 계정정보로 로그인이 잘 되는가?
    b. 잘못된 계정정보로 로그인이 실패하는가?
        i. 로그인 실패 여부가 “에러메세지" 로 확인이 가능하다면
        ii. 해당 항목은 확인 가능
4. **테스트를 위한 테스트를 하지 말 것.**

    실제로 사용자가 서비스를 사용할수 있도록 테스트를 해야함!

무엇을 테스트 할 것인가?

1. 다시 로그인으로 돌아가면
2. 로그인이 잘 되는지 확인하려면
    a. 이벤트 핸들러가 잘 동작하는지 봐야하고
    b. 결국은 버튼이 잘 그려지는지를 봐야한다.
3. 하지만 난 “로그인 성공" 이라는 간단한 기능에 유닛테스트를 하는 중인데?
    a. 그런데 로그인 성공을 테스트하려면
        i. 인풋도 잘 작동하고
        ii. 버튼도 잘 작동하고
        iii. HTTP request도 잘 이루어져야함
    b. 이정도면 통합테스트 아닌가?
        i. 프론트엔드는 유닛테스트와 통합테스트의 경계가 애매함 (개인적의견)

        버튼을 테스트 한다면/ 버튼에만 국한된 테스트가 아니고 => 다연결되어있기 때문이다.

어떻게 테스트 할 것인가?

1. Given - When - Then
2. True 또는 False를 리턴
3. 결과값을 expectation과 비교
4. 테스트는 한번에 하나씩만

단위테스트

1. 코드의 가장 작은 단위를 테스트
   a. 각각의 컴포넌트
   b. 각각의 함수
   c. Jest를 많이 사용함
2. 로그인으로 예를 들자면 cypress component testing => 추천안함
   a. 아이디와 비밀번호 입력 필드가 존재하는지.
   b. 아이디와 비밀번호 입력 필드가 적절한 속성(type, name 등)을 가지고 있는지
   c. 로그인 버튼이 존재하고, 활성 상태인지
   d. 아이디 또는 비밀번호가 없을 때 로그인 버튼을 클릭하면 에러 메시지가 표시되는지
   e. 로그인 함수(예: handleLogin())가 예상대로 동작하는지
3. 언급한 것처럼 단위테스트와 통합테스트의 경계는 모호함

통합테스트

1. 여러 컴포넌트나 모듈이 함께 작동하는 경우를 테스트
   a. 컴포넌트간 상호작용을 보는 것
   b. 단위테스트보다 넓은 범위
2. 로그인으로 예를 들자면
   a. 아이디와 비밀번호를 입력하고 로그인 버튼을 클릭했을 때, 올바른 요청이 서버로 전송되는지
   b. 서버에서 응답이 오면 적절하게 UI가 업데이트되는지
   c. 로그인이 성공한 후에 사용자를 적절한 페이지로 리디렉션하는지
3. 프론트엔드에서 단위테스트와 통합테스트의 경계가 애매함
   a. “모듈" 을 어떻게 정의할 것인지
   b. 프론트엔드에서 “기능"을 어떻게 정의할 것인지

cypress &&. jest 에서 에러나는 경우가 있음

cypress && jest 같이쓰려면 tsconfig 빡세게 줘야함 => expect 함수가 이름이 같은게 두개있음

cypress studio 를 함께 쓰면 더 좋음

E2E 테스트

목킹 : 서버로 요청이 들어가기전에 가로채서 정상처리가 된 척 하는것, 실제 시스템에서 잘 확인하기 힘든 예외처리 같은걸 확인하고 싶을때 모킹을 많이 함 (cy.intercept)

1. 사용자 관점에서 어플리케이션 플로우를 전체적으로 테스트
   a. 사용자의 실제 흐름을 테스트하는 것
2. 실제 환경에서 테스트
   a. 모든 컴포넌트
   b. 데이터베이스
   c. 네트워크
   d. 사용자 인터페이스
3. 로그인 기능을 예로 들면
   a. 사용자가 실제 환경과 같은 상황에서 웹사이트를 방문하여 아이디와 비밀번호를 입력하고, 로그인 버튼을 클릭하는 과정을 시뮬레이션
4. 애자일에 부적합하다고 생각함
   a. 빠른 대응/수정이 불가능함

프론트에 적합함

메서드 사용이나 및 구현이 유사하면, cypress 가 좀 더 전반적인 유저플로우를 테스트하기 쉽도록 툴이나 인터페이스가 잘 되어있는 것

결국엔 request 관련 테스팅을 할때는 jest로 고생하기 보다는 cypress를 쓰는 게 낫다.

Jest, ts 설정에 스트레스를 받으신다면 playwright도 추천

=> 작성을 다해야하기 때문

cypress 로짜고 => specs 에서 할수 있음

jest 는 node 에서 백엔드를 테스트하기 위해서 나온 것이기 때문에 jset 환경에 dom 이없음

fireEvent: 직접 해당 DOM 요소값을 조작하는데 반해

userEvent : 사용자 인터랙션을 입력하는 것

프론트엔드 TDD refactoring 할때가 테스트 코드 유용

1. 가능하긴함
    a. 디자인 시스템이 갖춰져있다면 특히 매우 수월함
    b. Atomic component 조합만 충분히 해도 가능
        i. data-testid, data-cy등등의 설정만 잘해주면 됨

프론트엔드 TDD

1. 가능하긴함
    a. 디자인 시스템이 갖춰져있다면 특히 매우 수월함
    b. Atomic component 조합만 충분히 해도 가능
        i. data-testid, data-cy등등의 설정만 잘해주면 됨
2. 하지만 꼭 해야하나?
    a. 애자일 조직 특성에 맞지 않음
    b. TDD를 해야만 검증이 되는건 아닌듯
        i. 만약 테스트코드를 잘못작성했다면?
        ii. 제대로된 검증은 어렵고, 코드상으로 리뷰만 해야하는데
        iii. 오히려 놓치는 부분이 나올 수 있음