Buduj Dappy za pomocą Angulara: część I

W poprzednich częściach omówiłem różne łańcuchy bloków i nauczyłeś się tworzyć inteligentne kontrakty, które mogą wchodzić w interakcje z łańcuchem bloków. Utworzyłeś inteligentne kontrakty w Ethereum, NEO, EOS i Hyperledger. W części 1 podzieliłem proces na pięć warstw: warstwę konsensusu, warstwę górniczą lub rezerwującą, warstwę propagacji, warstwę semantyczną i warstwę aplikacji. Inteligentne kontrakty są częścią warstwy aplikacji w cyklu rozwojowym; jednak warstwa aplikacji jest niekompletna bez interfejsu front-end, który umożliwia użytkownikowi końcowemu interakcję z łańcuchem bloków.

Wskazówka : Wiele razy usłyszysz zdecentralizowane aplikacje (dapps) nazywane inteligentnymi kontraktami. Inteligentne kontrakty to samowykonalne kontrakty. Dapps używają inteligentnych kontraktów, ale działają w sieci P2P i nie w jednym systemie.

Deweloperzy i bardziej doświadczeni użytkownicy mogą wchodzić w interakcję z inteligentnymi kontraktami, które utworzyłeś za pomocą interfejsu wiersza poleceń i narzędzi wymienionych w poprzednich rozdziałach, ale opracowanie aplikacji front-endowej, która jest w stanie wchodzić w interakcje z łańcuchem bloków, jest niezbędne dla wszystkich innych użytkowników. Robisz to, tworząc zdecentralizowaną aplikację (dapp). W tym i następnym rozdziale stworzysz zdecentralizowaną aplikację za pomocą Angulara, aby użytkownicy mogli wchodzić w interakcje z inteligentną umową za pomocą przyjaznego i intuicyjnego interfejsu użytkownika (UI). Podzieliłem ten proces na dwie części.

Część I, omówiona w tym rozdziale, obejmuje następujące tematy:

• Opracowanie dappa, w tym jego korzyści i klasyfikacji

• Korzystanie z Angular, w tym jego architektura, korzyści, wymagania wstępne i tworzenie aplikacji szkieletu Angular

• Tworzenie i stylizowanie niestandardowych komponentów Angular

Część II zawiera następujące tematy:

• Tworzenie inteligentnego kontraktu dapp z Truffle

• Integracja inteligentnego kontraktu z dapp

• Łączenie i łączenie aplikacji dapp z siecią Ethereum

Zacznijmy.

Co to jest Dapp?

Zdecentralizowana aplikacja (w skrócie ĐApp, dapp, Dapp, dApp lub DApp i wymawiana jako „dee-app”) to aplikacja internetowa, która może wchodzić w interakcje z inteligentną umową. Dappy działają na blockchainie i wykorzystują rozproszoną księgę. Blockchain Ethereum jest obecnie najpopularniejszą platformą do uruchamiania dappów; jednak inne technologie rozproszonej księgi (DLT), które widziałeś, również zapewniają możliwość tworzenia dappów. W poprzednich rozdziałach omówiłem NEO, EOS i Hyperledger; inne to ICON, Cardano i Hashgraph (Hedera).

„Wszystko, co można zdecentralizować, zostanie zdecentralizowane”. - David Johnston, dyrektor generalny DApp Fund

Jeśli kiedykolwiek tworzyłeś standardową aplikację komputerową, internetową lub mobilną, przekonasz się, że dapps są podobne, ale także bardzo różne. Dapp jest tworzony przy użyciu tych samych narzędzi i języków, których używasz do tworzenia dowolnej innej aplikacji, ale aby aplikacja została zaklasyfikowana jako dapp, musi spełniać następujące kryteria:

- Open source: jego kod jest publikowany jako open source i nie powinien być zarządzany przez jeden podmiot (scentralizowany). Należy pamiętać, że aplikacja może dostosować swój własny protokół w odpowiedzi na proponowane ulepszenia i informacje zwrotne z rynku; jednak konsensus jego użytkowników napędza wszystkie zmiany.

- Zdecentralizowany: Dapps wykorzystują łańcuch bloków lub sieć P2P.

- Zachęta: Dapps wykorzystują zasoby cyfrowe do finansowania.

- Algorytm/protokół: Dappy często generują tokeny i zawierają mechanizm konsensusu, taki jak PoW, PoS, a nawet własny.v

Te kryteria zapewniają, że dapps nie mają przestojów, jak inne aplikacje pobierane z rynków, takich jak iTunes lub Google Play; dapps dają również kontrolę społeczności zamiast jednej jednostce. Te kryteria mogą być znaczące. Na przykład Apple i Google często odrzucają aplikacje, ponieważ nie spełniają ich arbitralnych lub opartych na pieniądzach zasad. Zasady te nie zawsze mają sens i nie zawsze leżą w najlepszym interesie użytkownika końcowego; często są po to, aby blokować wykorzystanie konkurenta lub dla zysku pieniężnego. Dappy, które są oparte na kodzie open source zaimplementowanym na zdecentralizowanych łańcuchach bloków i finansowane z tokenów generowanych przy użyciu określonego mechanizmu konsensusu, są uważane przez wielu za przyszłość wszystkich firm. Tylko czas powie. Dodatkowo oprogramowanie open source jest zaletą, ponieważ umożliwia użytkownikom przeglądanie kodu źródłowego i potencjalny wkład. Decentralizacja przy użyciu blockchain wykorzystuje zalety blockchain jako DLT i służy jako zamiennik tradycyjnej jednoserwerowej bazy danych. Wreszcie, dodawanie rekordów/transakcji do ksiąg odbywa się zwykle za pomocą tokenów, a mechanizm konsensusu tokena jest również umową między wszystkimi użytkownikami dappa.

Klasyfikacja Dappa

Oprócz poprzednich kryteriów, dapps można kategoryzować. Klasyfikacja opiera się na infrastrukturze wykorzystywanej przez dapp i można ją podzielić na trzy kategorie:

- Dedykowane dappy łańcucha bloków: są to dappy, które bezpośrednio korzystają z dedykowanego łańcucha bloków; przykładami są bitcoin, Ethereum, EOS i NEO.

- Dappy opierające się na innym łańcuchu bloków: Na przykład protokół Omni Layer (wcześniej nazywany Mastercoin) to cyfrowa waluta i protokół komunikacyjny, który jest zbudowany na łańcuchu bloków bitcoin.

- Dappy opierające się na innym protokole zbudowanym na innym blockchainie: Te dappy używają protokołu zbudowanego na innym blockchainie. Przykładem jest bezpieczna sieć wykorzystująca protokół Omni Layer.

Dobrym przykładem pomagającym zrozumieć koncepcję klasyfikacji jest token USDT (Tether). Token ten został wyemitowany dwukrotnie w oparciu o dwa blockchainy: bitcoin i Ethereum. W tym przypadku istnieją dwa rodzaje USDT. Oryginał, który jest oparty na bitcoinie, odbywa się za pomocą protokołu Omni Layer do generowania tokena, a USDT oparty na Ethereum jest zgodny ze standardem Ethereum ERC20.

Projekty Dapp

Większość dappów jest budowana bezpośrednio na łańcuchu blokowym Ethereum lub używa łańcucha blokowego dla swoich tokenów. Istnieje jednak kilka dappów, które nawet budują własny dedykowany blockchain. Spójrz na tabelę, aby zobaczyć próbkę różnych dappów i ich klasyfikacje.

Dapp : Opis : Klasyfikacja : Token : Blockchain

Ethlance : Giełda ogłoszeń o pracę i zatrudniania freelancerów. 0 procent opłat. : Używa bezpośrednio Ethereum : Brak tokena : Ethereum blockchain

Golem : Globalny rynek bezczynnej mocy komputerowej. : Token oparty na Ethereum : GNT: Ethereum blockchain

Sieć SAFE : Sieć przechowywania danych i komunikacja. : Implementacja oparta na innym protokole (protokół Omni), który jest zbudowany na innym blockchainie (bitcoin) : SFE : Bitcoin

Jak stworzyć własny Dapp?

Sukces bitcoina i blockchain przyniósł eksplozję dappów. Deweloperzy i właściciele firm stworzyli podstawowy proces tworzenia dappów. Nie musisz tego dokładnie przestrzegać i może się to zmienić do czasu pisania; jednak wiele opublikowanych dappów podążyło za tym procesem. Proces składa się z tych pięciu kroków:

1. Napisz białą księgę.

2. Uruchom pierwszą ofertę monet (ICO).

3. Opracuj dapp.

4. Uruchom swój dapp.

5. Sprzedaj swój dapp.

Przyjrzyjmy się tym krokom.

Napisz białą księgę

Biała księga jest podobna do biznesplanu firmy skierowanego do inwestorów. Jednak jest skierowany nie tylko do inwestorów; to plan techniczny. Biała księga jest dokumentem technicznym, a także biznesplanem i powinna wyjaśniać rozwiązywany problem oraz koncepcję, funkcje i aspekty techniczne dappa. Podobnie jak w biznesplanie, dobrym pomysłem jest dołączenie unikalnej propozycji sprzedaży (USP), mapy drogowej, życiorysów członków, możliwości i historii, aby pomóc w ustaleniu wiarygodności.

Uwaga: Unikalna propozycja sprzedaży (USP) to problem, który Twój dapp ma rozwiązać.

Po opublikowaniu białej księgi dobrze jest uzyskać informacje zwrotne od rówieśników i społeczności na wczesnych etapach i przed opracowaniem. Media społecznościowe, formularze i publikacje są często wykorzystywane do promowania aplikacji i pomagania w budowaniu wiarygodności.

Uruchom początkową ofertę monet

Po opublikowaniu białej księgi następnym krokiem jest uruchomienie ICO i sprzedaż monet lub tokenów w celu sfinansowania i wsparcia Twojego dapp. Moneta powinna mieć powód istnienia, a nie być taka sama jak inna moneta/token, więc powinieneś wyjaśnić, w jaki sposób i dlaczego twój dapp

potrzebuje własnego tokena lub monety. Musisz również zdecydować o rodzaju klasyfikacji dla swojego dappa, który określi, czy będziesz potrzebować jednego lub wszystkich z poniższych: 1) wystawić token 2) ustawić opłaty za użytkowanie. 3) posiadać dedykowany blockchain. 4) posiadać mechanizm wydobywczy 5) ustawić alokację opłat 6) nagradzać inwestorów 7) przydzielać opłaty na różne działy Twojej firmy: wsparcie, rozwój, marketing i biznes.

Opracuj Dapp

Programowanie powinno być open source, a GitHub jest zwykle używany do repozytoriów do prac programistycznych. W każdym wydaniu dobrym pomysłem jest poinformowanie inwestorów i innych o wydaniu, aby budować użytkowników i społeczność deweloperów wokół swojego projektu. Wiele dappów próbowało zdobyć fundusze i nie dostarczyło żadnych użytecznych produktów; wyróżnij się i unikaj potencjalnych problemów z regulatorami.

Uruchom swój Dapp

Uruchom swój dapp i dołącz informacje o wersji, dokumentację, mapę drogową i plan konserwacji. Bardzo ważne jest, aby dotrzymać obiecanej daty premiery.

Sprzedaj swój Dapp

Ostatnim krokiem jest marketing. Oprócz tradycyjnego marketingu, dappy często zatrudniają lub pracują z suflerami we wczesnych fazach lub po wydaniu, aby przekazać wiadomość. Innym unikalnym aspektem marketingowym dappa jest umieszczenie monety/tokenu na giełdach. To jest ostatnia pieczęć uznania. Niektóre giełdy mają wdrożony system głosowania, aby wybrać następną monetę/żeton do wystawienia. Niektóre giełdy nadużywają tego procesu i pobierają wysokie opłaty za wystawienie tokena lub monety. Na przykład lista tokenów użytkowych na giełdzie Binance może kosztować od 0,5 miliona do 3 milionów dolarów. Wielu wczesnych inwestorów, w tym właścicieli dappów, było w stanie „wypłacić", jeśli token jest notowany na głównych giełdach, ponieważ jego cena często rośnie z powodu notowania; Jednak coraz trudniej jest umieścić dapp na liście i musi zapewniać prawdziwą wartość. Oszuści są często ujawniani, a monety/tokeny są usuwane z listy tak szybko, jak są wymienione.

Dlaczego Agular?

Dzięki dapps, tak jak w przypadku każdej tradycyjnej aplikacji, możesz pisać swoją aplikację natywnie na urządzeniu, na którym publikujesz swoją aplikację (w obsługiwanym języku urządzenia, takim jak Xcode dla iOS); jednak udowodniono, że korzystanie z frameworka może przyspieszyć rozwój. Na przykład, jeśli chcesz wykorzystać ten sam kod i wdrożyć aplikację na wielu urządzeniach o różnych rozmiarach ekranu, może to stać się wyzwaniem dla małego zespołu. Angular pomaga tworzyć wieloplatformowe nowoczesne aplikacje internetowe, mobilne i pulpit w tym samym czasie. Angular CLI i Component Dev Kit (CDK) mogą przyspieszyć tworzenie aplikacji. Korzystanie z Angulara może być korzystne ze względu na następujące czynniki:

- Duże wsparcie społeczności

- Architektura korporacyjna i skalowanie

- Obsługa wielu platform

- Dokumentacja

Angular jest strukturą strukturalną i umożliwia tworzenie aplikacji front-end po stronie klienta. Kawałki są luźno połączone i ustrukturyzowane w sposób modułowy, co skutkuje mniejszą ilością kodu do

pisania, większą elastycznością, łatwiejszym do odczytania kodem i szybszym czasem programowania. Angular pozwala programiście stworzyć zestaw narzędzi do budowania framework, który będzie dokładnie pasował do potrzeb Twojej aplikacji. Możesz użyć HTML jako języka szablonu i rozszerzyć składnię HTML, aby składniki aplikacji można było łatwo odczytać. Poza HTML, kodowanie odbywa się za pomocą TypeScript, który przekształca JavaScript w język programowania obiektowego i zapewnia środowisko na poziomie przedsiębiorstwa. Ponadto Angular jest dobrze skonstruowany i zbudowany tak, aby był w pełni dostępny, zgodnie z dostępnymi bogatymi aplikacjami internetowymi (ARIA), dzięki czemu Twoja aplikacja lub witryna może być poprawnie zbudowana dla osób niepełnosprawnych. Angular dobrze dogaduje się również z innymi bibliotekami JavaScript, dzięki czemu można zainstalować takie biblioteki, jak Ethereum JavaScript API web3.js z menedżerem npm. Wreszcie, funkcje Angulara można łatwo modyfikować lub wymieniać, aby dopasować je do Twoich potrzeb.

Uwaga: Słowo angular oznacza posiadanie wielu kątów lub mierzone pod kątem. Angular jest strukturą strukturalną i umożliwia tworzenie aplikacji front-end po stronie klienta dla sieci Web, urządzeń mobilnych i komputerów stacjonarnych. Jest to open source, front-end framework do dynamicznego tworzenia aplikacji.

Najważniejsze cechy Angulara to wiązanie danych i wstrzykiwanie zależności. Mogą one pomóc w skróceniu kodu. Ponadto Angular istnieje od lat; jest w siódmym wydaniu.

Uwaga : Wstrzykiwanie zależności to technika wzorca projektowego. Jak sama nazwa wskazuje, oznacza to użycie jednego obiektu jako zależności do innego obiektu poprzez wstrzyknięcie kodu.

Wskazówka: wybrałem Angular, ale Angular nie jest jedynym frameworkiem, który może przyspieszyć rozwój. Możesz skorzystać z innych frameworków, takich jak React i osiągnąć podobne korzyści. Ta decyzja jest tak naprawdę kwestią osobistego gustu i umiejętności zespołu. Możesz łatwo przekonwertować ten projekt na projekt React, głównie kopiując pliki swojego projektu do projektu React.

Tworzenie Agular Dapp

W tej sekcji utworzysz rzeczywisty dapp, który połączy się z siecią Ethereum i przeniesie środki z jednego konta na drugie. Jest to często podstawowa funkcja każdego dappa. Na przykład możesz zbudować dapp, który sprzedaje produkty, świadczy usługi lub płaci użytkownikom za wypełnianie quizów, a wszystkie te typy muszą mieć mechanizm przesyłania monet/tokenów. W tej części tego rozdziału będziesz tworzyć dapp z wykorzystaniem Angulara. Jeśli chodzi o środowisko i wdrożenie, będziesz korzystać z frameworka internetowego Truffle, którego użyłeś w części 5, ponieważ oferuje on korzyści w zakresie szybkiego tworzenia inteligentnego kontraktu. Truffle jest w stanie zrobić więcej niż tylko pomóc w skompilowaniu inteligentnego kontraktu; robi wszystko, czego potrzebujesz, aby wstrzyknąć inteligentną umowę do aplikacji internetowej i może uruchomić zestaw testowy. Zamierzasz również ponownie użyć MetaMask, aby uzyskać bezpieczne konto blockchain w przeglądarce. Na koniec użyjesz i uruchomisz Ganache, aby utworzyć lokalny serwer RPC blockchain do testowania i programowania.

Warunki wstępne

Większość tego, czego potrzebujesz, jest już zainstalowana. Angular potrzebuje menedżera węzłów i npm, które zainstalowałeś wcześniej. Sprawdź, czy zainstalowana jest poprawna wersja, uruchamiając biblioteki z flagą v.

```
> node -v
```

> npm -v

Jeśli nie masz npm i węzła, po prostu uruchom następujące polecenie:

> brew install node

Przyznaj własność npm swojemu użytkownikowi, aby nie trzeba było używać sudo do instalowania bibliotek.

> sudo chown -R $USER:$GROUP ~/.npm

> sudo chown -R $USER:$GROUP ~/.config

Zaleca się uaktualnienie npm, aby upewnić się, że korzystasz z ostatniej wersji.

> [sudo] npm install -g npm

+npm@6.9.0

Angular CLI

Następnie musisz zainstalować interfejs wiersza poleceń Angular (CLI). W przypadku Angular CLI zaleca się (ale nie jest to wymagane) zainstalowanie Angular CLI z sudo i allow-root oraz upewnienie się, że Angular CLI będzie miał odpowiednie uprawnienia. Będziesz instalować wersję 14, która jest najnowszą stabilną wersją Angulara.

> sudo npm install -g @angular/cli@14.0.0--unsafe-perm=true

--allow-root

+ @angular/cli@14.0.0

added 363 packages from 197 contributors in 13.691s

Możesz także zainstalować najnowszą wersję Angulara, ale Twój przykładowy kod może się zepsuć z nowszymi wersjami Angulara.

> sudo npm install -g @angular/cli --unsafe-perm=true --allow-root

Aby sprawdzić, czy instalacja poszła dobrze, uruchom flagę wersji i powinieneś zobaczyć wersję 7.3.9; Rysunek 9-2 przedstawia oczekiwany wynik.

> ng version

Utwórz projekt Agular

Teraz, gdy masz zainstalowane główne narzędzia i biblioteki, możesz kontynuować i tworzyć swój projekt od podstaw, pobierając inne potrzebne biblioteki, biblioteki testowe i skrypty kompilacji, a także tworząc własną strukturę folderów; jednak, aby przyspieszyć ten proces, możesz użyć projektu źródłowego Angular, który zawiera projekt szkieletowy, aby szybko załadować swój projekt. Korzystanie z projektu nasion Angulara może pomóc w szybkim i wydajnym rozpoczęciu rozwoju, zgodnie z najlepszymi praktykami Angulara. Istnieją plusy i minusy korzystania z kodu szkieletowego. Możesz sam zdecydować, czy chcesz użyć tego szkieletu w przyszłych projektach, ale dla tej aplikacji demonstracyjnej jest idealny. Istnieje wiele sposobów tworzenia projektu za pomocą szkieletu nasion Angulara. Pokażę Ci tutaj dwie opcje: użycie Angular CLI i użycie WebStorm. Nowe polecenie ng uruchomi skrypt, który utworzy Twoją aplikację. Możesz uruchomić nowe polecenie CLI i podać nazwę ethdapp jako nazwę swojej aplikacji.

> cd ~/desktop

> ng new ethdapp

Would you like to add Angular routing? (y/N) y

Which stylesheet format would you like to use? CSS

Zauważ, że dodałem tutaj routing i zdecydowałem się użyć CSS do stylów. Zajmę się nimi w dalszej części. Po zakończeniu instalacji wyświetli wszystkie utworzone pliki.

CREATE ethdapp/README.md (1024 bytes)

CREATE ethdapp/angular.json (3557 bytes)

CREATE ethdapp/package.json (1313 bytes)

&helip;

Zmień katalogi na nowo utworzony folder i potwierdź, że masz początkowe pliki i katalogi.

> cd ethdapp

Uruchomienie następującego polecenia spowoduje przeanalizowanie pliku konfiguracyjnego package.json z zaleceniami:

> ng update

Możesz uruchomić następujące polecenie, aby postępować zgodnie z zaleceniami:

> ng update –all

Następnie zainstaluj Bower globalnie. Bower to menedżer pakietów, który jest często używany z Angularem.

> npm install -g bower

> bower -v

1.8.8

Zróbmy przewodnik po tym, co zostało utworzone w obszarze roboczym i plikach projektu startowego

- Nowy obszar roboczy: jest to folder główny o nazwie ethdapp.

- Folder e2e: Zawiera kompleksowy projekt testowy, zlokalizowany tutaj: ethdapp/e2e. Folder testowy zawiera plik konfiguracyjny JSON biblioteki Jasmin.

- src folder: To jest folder twojego projektu, który zawiera wszystkie pliki twojego projektu.

• Wstępny projekt szkieletowej aplikacji, znajdujący się tutaj: ethdapp/src/app

• Folder zasobów z plikiem wpisu index.html

• Inne pliki konfiguracyjne

- .gitignore: tutaj wymieniasz wszystkie pliki i foldery, które chcesz zignorować podczas przesyłania projektu do Git.

- angular.json: To jest plik konfiguracyjny twojego projektu i zawiera informacje o twoim projekcie.

- package.json: jest to plik konfiguracyjny menedżera npm i zawiera wszystkie biblioteki, których będziesz używać w swoim projekcie.

- README.MD: To jest dokumentacja dotycząca twojego projektu; będzie to dokument „strony głównej" twojego projektu i pierwsi programiści plików przeczytają, aby uzyskać instrukcje, jak uruchomić projekt.

- tsconfig.json: to jest plik konfiguracyjny TypeScript.

- tslint.json: jest to plik konfiguracyjny Lint używany do ustawienia najlepszego formatowania, odstępów i tym podobnych.

Obsługuj aplikację

Aby zobaczyć swoje rzeczywiste dapps, będziesz używać polecenia ng serve, które kompiluje aplikację, uruchamia serwer deweloperski, obserwuje pliki źródłowe i odbudowuje aplikację podczas wprowadzania zmian w tych plikach. Flaga --open otwiera aplikację w przeglądarce na porcie 4200 tutaj: http://localhost:4200/. Uruchom polecenie ng serve z flagą open.

> ng serve --open

Powinieneś zobaczyć działający dapp w przeglądarce.

Aplikacja szkieletowa zawiera linki do wycieczki, dokumentacji i bloga Angulara. Przechodząc przez „Tour of Heroes" i dokumentację CLI, możesz dobrze zrozumieć, jak działa Angular, a dodawanie zakładek do bloga Angulara może dać ci aktualizacje dotyczące przyszłych wydań i ogłoszeń. Aby zatrzymać udostępnianie aplikacji, naciśnij Command + C w Terminalu.

Projekt Angular z WebStorm

Inną opcją uruchomienia projektu Angular seed jest wykorzystanie WebStorm IDE, którego używałeś w poprzednich częściach. WebStorm umożliwia zaimportowanie utworzonego projektu źródłowego lub utworzenie nowego projektu źródłowego. Aby zaimportować projekt ethdapp utworzony za pomocą polecenia Angular CLI ng new, otwórz WebStorm, wybierz Plik -> Otwórz i przejdź do katalogu ethdapp. Otóż to; WebStorm automatycznie zaimportuje projekt. Alternatywnie, aby rozpocząć nowy projekt Angular seed w WebStorm, wybierz File -> New -> Project z górnego menu. Następnie wybierz Angular CLI i nazwij swój projekt ethdapp. Użyj menu rozwijanego, aby wybrać wersję interfejsu Angular CLI. Po utworzeniu projektu możesz uruchomić to samo polecenie, zmieniając kartę Terminal w dolnym menu WebStorm.

> ng serve –open

Po pobraniu kroków upewnij się, że uruchamiasz npm install, ponieważ usunąłem moduł węzła, aby zmniejszyć rozmiar projektu.

> npm install

Uwaga : Wykluczyłem z projektu node_modules, który zawiera wszystkie zależności projektu. Często nie dołącza się go do projektu ze względu na jego wielkość; możesz go zainstalować za pomocą polecenia npm install.

Zapewnij brak niezgodności z wersją Angular CLI

Możesz utworzyć projekt źródłowy Angular za pomocą WebStorm lub za pomocą polecenia ng i musisz sprawdzić, czy nie ma niezgodności globalnego interfejsu Angular CLI z lokalnym interfejsem Angular

CLI projektu. Może się to zdarzyć podczas ustawiania plików wskazujących na poprzednią wersję lub mogłeś używać Angular w przeszłości ze starszą wersją. Dzieje się tak, że twój lokalny projekt Angular pokazuje starszą wersję niż globalny Angular zainstalowany na twoim komputerze. Aby upewnić się, że tak nie jest, uruchom dowolne polecenie ng, a jeśli ten problem występuje, zobaczysz następujący komunikat o błędzie:

> ng

Twoja globalna wersja Angular CLI (7.3.9) jest wyższa niż wersja lokalna (6.2.9). Używana jest lokalna wersja Angular CLI. Jeśli będziesz kontynuować te ustawienia, będziesz korzystać z wersji 6.x zamiast 7.x. Aby to naprawić, musisz odinstalować Angular CLI ze środowiska deweloperskiego, a następnie zainstalować wersję 7.x.

> npm uninstall --save-dev angular-cli

> npm install --save-dev @angular/cli@7.3.9

Zwróć uwagę, że używasz flagi --save-dev, aby nowa wersja została zapisana w pliku projektu package.json. Teraz, jeśli ponownie uruchomisz polecenie version, powinieneś zobaczyć poprawną wersję bez komunikatów ostrzegawczych.

> ng --version

Angular CLI: 7.3.2

Teraz, gdy masz pewność, że korzystasz z właściwej wersji Angular CLI, jesteś gotowy do dalszego rozwoju i wprowadzania zmian w aplikacji startowej nasion.

Komponenty Angular

Najlepszą praktyką Angulara jest użycie architektury w stylu kontrolera widoku modelu (MVC). Angular obsługuje kodowanie z oddzieleniem problemów, tak jak każdy inny dojrzały framework. Angular MVC zawiera następujące trzy elementy:

- Model: zawiera dane aplikacji i powiązanie danych Angular, które umożliwia odzwierciedlenie danych.

- Widok: zawiera kod HTML lub szablon i dyrektywy.

- Kontroler: jest to klej łączący model i widok. Kontroler pobiera dane, stosuje logikę biznesową i wysyła wyniki do widoku.

Uwaga : odbicie w odniesieniu do powiązania danych, elementy powiązane z danymi oraz wszelkie zmiany danych są automatycznie odzwierciedlane. Na przykład powiążesz zmianę ceny z wieloma elementami widoku, a po zaktualizowaniu danych o zmianie ceny wszystkie elementy widoku zostaną zaktualizowane automatycznie.

Jak zapewne pamiętasz, strona powitalna Angulara otworzyła się, gdy uruchomiłeś komendę serwowania. Elementem powitalnym jest powłoka aplikacji. Powłoka jest kontrolowana przez składnik Angular o nazwie AppComponent. Komponenty są podstawowymi blokami konstrukcyjnymi aplikacji Angular. Wyświetlają dane na ekranie, nasłuchują danych wejściowych użytkownika i podejmują działania w oparciu o te dane wejściowe. Stworzysz komponent o nazwie transfer, którego będziesz używać do przesyłania monet na inny adres. Aby utworzyć komponent transferu, uruchom polecenie ng generate component.

> ng g c components/transfer

Zauważ, że użyłeś skrótów g i c, które oznaczają „generowanie” i „komponent”, odpowiednio, ale możesz również użyć pełnej nazwy zamiast skrótu. Polecenie ng wygenerowało dla Ciebie cztery następujące pliki:

- transfer.component.css: specyficzne style CSS komponentu- transfer.component.html: Szablon komponentu napisany w HTML

- transfer.component.spec.ts: Plik testowy

- transfer.component.ts: Kod klasy komponentu napisany w TypeScript

Te cztery pliki razem działają jako implementacja komponentu transfer.

Struktura aplikacji jest zwykle tworzona z nagłówkiem, stopką i menu nawigacyjnym, dzięki czemu można przechodzić do różnych widoków częściowych. Korzystanie z tej architektury komponentów nagłówka i stopki może pomóc w tworzeniu różnych widoków i dzieleniu widoku strony na osobne pliki. Pomyśl o każdym kawałku jako samodzielnym module wielokrotnego użytku. Angular Seed promuje ten typ architektury i jest dostarczany z już utworzonym komponentem powitalnym. Stwórzmy składnik początkowy, nagłówek i stopkę.

> ng g c components/start

> ng g c components/header

> ng g c components/footer

Możesz zobaczyć w danych wyjściowych, że każdy komponent wygenerował następujące pliki:

CREATE src/app/components/[component-name]/[component-name].component.css

CREATE src/app/components/[component-name]/[component-name].component.html

CREATE src/app/components/[component-name]/[component-name].component.spec.ts

CREATE src/app/components/[component-name]/[component-name].component.ts

Oprócz tych plików możesz otworzyć ethdapp/srcapp/app.module.ts i zauważ, że plik app.module.ts był modyfikowany za każdym razem, gdy tworzyłeś komponent. Plik app.module.ts jest jednym z najważniejszych plików w Angularze; to kontroler aplikacji napisany w TypeScript. Kontroler jest plikiem globalnym, który powiąże ze sobą komponenty, więc każdy komponent, którego chcesz użyć w swojej aplikacji, musi być zdefiniowany w tym pliku. Jeśli nie używałeś skryptu ng, będziesz musiał samodzielnie zmodyfikować app.module.ts, aby połączyć się z nowym komponentem. Ponieważ korzystałeś z CLI, te importy są uwzględniane automatycznie:

import { TransferComponent } from './components/transfer/

transfer.component';

import { StartComponent } from './components/start/start.component';

import { HeaderComponent } from './components/header/header.component';

import { FooterComponent } from './components/footer/footer.component';

Moduł Routing

Innym ważnym plikiem i dobrą praktyką do tworzenia jest moduł app-routing. Ten plik działa jako kontroler, który instruuje Angular, jak nawigować do różnych widoków w Twojej aplikacji. Zwykle, aby wygenerować trasę dla swojej aplikacji, nie musisz tego robić ręcznie, ponieważ podczas tworzenia aplikacji zdecydowałeś się utworzyć plik routingu o nazwie app-routing. Jeśli chcesz utworzyć plik app-routing, możesz uruchomić następujące polecenie module:

> ng generate module app-routing --flat --module=app

CREATE src/app/app-routing.module.ts

UPDATE src/app/app.module.ts

Zauważ, że tym razem w swoim poleceniu używasz modułu generowania pełnych nazw zamiast tylko pierwszych liter g i m. Obie opcje działają w ten sam sposób. Polecenie generate module tworzy początkowy kod pokazany na listingu dla src/app/app-routing.module.ts.

Listing app-routing Initial Startup Code

```
import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }
```

Początkowy kod zawiera instrukcję importu do kodu Angular i tag modułu. Następnie zastąp wstępnie wypełniony kod app-routing. module.ts kodem z listingu 2.

Listing 2. app-routing Code to Route Views

```
import { NgModule } from '@angular/core';

import { CommonModule } from '@angular/common';

import { RouterModule, Routes } from '@angular/router';

import { StartComponent } from './components/start/start.

component';

import { TransferComponent } from './components/transfer/

transfer.component';

const routes: Routes = [

{ path: '', redirectTo: '/start', pathMatch: 'full' },

{ path: 'start', component: StartComponent },
```

```
{ path: 'transfer', component: TransferComponent }

];

@NgModule({

declarations: [],

imports: [ RouterModule.forRoot(routes), CommonModule ],

exports: [ RouterModule ]

})

export class AppRoutingModule { }
```

Na listingu 2 zaimportowałeś komponenty widoku, których będziesz używać; są to start i transfer. Będą działać jako strony internetowe w witrynie internetowej lub częściowe widoki w aplikacji mobilnej. Trasa informuje aplikację, który widok ma pasować do danego słowa kluczowego, a na koniec ustawiasz instrukcje importu, aby poinformować Angular, kto może uzyskać dostęp do tego modułu. Teraz, gdy trasowanie jest ustawione, możesz wyświetlić stopkę, nagłówek i treść strony. Wszystko, co musisz zrobić, to otworzyć src/app/app.component.html i zaktualizuj kod HTML strony powitalnej do następujących trzech wierszy:

```
< app-header >< /app-header >

< router-outlet >< /router-outlet >

< app-footer >< /app-footer >
```

Aby przetestować zmiany wprowadzone w aplikacji, nie musisz ponownie opublikuj swoją aplikację lub uruchom dowolne skrypty; po prostu zapisz pliki i uruchom to samo polecenie obsługi, które uruchomiłeś wcześniej w Terminalu.

```
> ng serve

[wdm]: Compiled successfully.
```

Skrypt udostępniania zawiera skrypty, które kontrolują zmiany w plikach i automatycznie aktualizują aplikację, więc po wprowadzeniu zmian w plikach wystarczy wrócić do przeglądarki. W większości przypadków nie musisz nawet odświeżać swojej strony internetowej; zmiany pojawią się tam automatycznie. Przejdź do http://localhost:4200, aby zobaczyć zmiany. Jeśli chcesz przejść bezpośrednio do strony transferu, wystarczy dodać słowo kluczowe wybrane na końcu adresu URL podczas konfigurowania mechanizmu routingu: http://localhost:4200/transfer.

Stylizacja aplikacji Angular

Twoja aplikacja w tym momencie nie ma stylu i wyświetla tylko tekst z nagłówkiem, stroną i stopką; jednak zanim zaczniesz stylizować, pomocne jest zrozumienie architektury stylu Angular, aby upewnić się, że nie skończysz z plikiem kaskadowych arkuszy stylów (CSS), który jest zbyt duży, aby można było nim zarządzać. Możesz stylizować swoją aplikację na poziomie globalnym, używając stylów, których potrzebujesz w całej aplikacji, a także określonego stylu unikalnego tylko dla jednego składnika. Dodatkowo fajnie byłoby szybko sprintować od zera do stylizowanej aplikacji. Można to zrobić za pomocą materiału kątowego. Angular Material zapewnia skrót do spójnego „wyglądu” aplikacji bez konieczności myślenia o programowaniu w różnych przeglądarkach i na różnych urządzeniach. Spójrzmy.

Architektura w stylu Angular

Angular jest skonfigurowany tak, aby mieć globalny plik CSS. Ten plik CSS nazywa się stylem. css i możesz go znaleźć w katalogu głównym projektu. src/style.css zawiera style, których chcesz użyć dla całej aplikacji, takie jak czcionki, motywy, style dla wszystkich składników i tak dalej. Jak widać, każdy składnik zawiera również prywatny plik CSS. Specyficzny plik CSS komponentu to miejsce, w którym umieszczasz style, które są unikalne i używane tylko dla tego komponentu. Na przykład /src/app/components/footer/footer.component.css zawiera style specyficzne dla komponentu stopki.

Materiał Angular

W tej chwili Twoja aplikacja startowa jest szybka, ponieważ zawiera minimalny kod; istnieje jednak potencjalny problem z wydajnością w miarę dodawania do aplikacji coraz większej liczby składników, zasobów i stylu. Możesz łatwo załadować swoją aplikację, a każda milisekunda się liczy. Innym potencjalnym problemem jest testowanie. Wszystkie różne przeglądarki, wersje przeglądarek, rozmiary ekranu i urządzenia muszą zostać przetestowane, a tworzenie stron od podstaw będzie wymagało rygorystycznych testów i zespołu ds. zapewnienia jakości (QA), aby zapewnić spójność działania na różnych urządzeniach. Angular Material rozwiązuje wszystkie te problemy oraz zapewnia dostępność i internacjonalizację. Dzieje się tak, ponieważ Angular Material jest zoptymalizowany pod kątem Angular i zbudowany przez zespół Angular, więc bezproblemowo integruje się z Angularem. Przeszedł już wszystkie te testy kompatybilności.

Zainstaluj Material Angular

Material można zainstalować na kilka sposobów. Ponieważ zainstalowałeś Angular DevKit, możesz po prostu uruchomić polecenie ng add, aby uzyskać bibliotekę Angular Material. Musisz najpierw zainstalować cdk, ponieważ jest to zależność.

> ng add @angular/cdk

Następnie zainstaluj materiał.

> ng add @angular/material

Zauważ, że dane wyjściowe pytają, jaki kolor motywu chcesz z linkami. Tematy omówię w następnej części , ale na razie wybierz pierwszy lub dowolny kolor, który wolisz.

? Choose a prebuilt theme name, or "custom" for a custom theme:

(Use arrow keys)

□ Indigo/Pink [Preview: https://material.angular.

io?theme=indigo-pink]

Deep Purple/Amber [Preview: https://material.angular.

io?theme=deeppurple-amber]

Pink/Blue Grey [Preview: https://material.angular.

io?theme=pink-bluegrey]

Purple/Green [Preview: https://material.angular.

io?theme=purple-green]

Możesz także skonfigurować rozpoznawanie gestów i animacje.

? Set up HammerJS for gesture recognition? Yes

? Set up browser animations for Angular Material? Yes

Oczekiwany wynik powinien pokazywać pliki, które zostały zaktualizowane:

UPDATE package.json

UPDATE angular.json

UPDATE src/app/app.module.ts

UPDATE src/index.html

UPDATE src/styles.css

Importuj moduły materiałów kątowych

Następnie chcesz zmodyfikować swoją aplikację, aby Angular Material zawierał animacje, ikony materiałów, obsługę gestów i moduły komponentów.

Importuj moduły Material Angulr

Następnie chcesz zmodyfikować swoją aplikację, aby Angular Material zawierał animacje, ikony materiałów, obsługę gestów i moduły komponentów. W swoim projekcie będziesz używać tylko modułów komponentów, a nie wszystkich funkcji, które ma do zaoferowania Angular Material; co musisz zrobić, to zaimportować NgModule dla każdego komponentu, którego chcesz użyć. Otwórz src/app/app.module.ts i dodaj instrukcje importu.

import {

MatButtonModule,

MatCheckboxModule,

MatInputModule,

MatSelectModule,

MatDatepickerModule,

MatNativeDateModule

} from '@angular/material';

Następnie zaktualizuj instrukcje importu @NgModule, aby uwzględnić zaimportowane moduły materiałów.

imports: [

BrowserModule,

AppRoutingModule,

BrowserAnimationsModule,

MatButtonModule,

MatInputModule,

MatDatepickerModule,

MatNativeDateModule,

MatCheckboxModule,

MatSelectModule

]

Otóż to. Możesz teraz mieć dostęp do dołączonych komponentów Angular Material.

Motywuj swoją aplikację do Angular Material

Teraz, gdy masz dostęp do komponentów Angular Material, możesz użyć motywów do ich stylizacji. Motyw to zestaw kolorów, które będą używane w komponentach Angular Material. W Angular Materi

– Paleta podstawowa: są to kolory najczęściej używane na wszystkich ekranach i komponentach.

- Paleta akcentów: są to kolory używane dla przycisku i elementów interaktywnych.

- Paleta ostrzegania: są to kolory błędów.

- Paleta pierwszego planu: są to kolory tekstu i ikon.

- Paleta tła: są to kolory tła elementu.

W Angular Material wszystkie style motywów są generowane statycznie podczas czasu kompilacji, aby uniknąć spowolnienia aplikacji podczas uruchamiania. Angular Material jest dostarczany z kilkoma gotowymi plikami CSS z motywami. Jak zapewne pamiętasz, miałeś możliwość wybrania motywu do użycia podczas instalacji Materiału. Te pliki motywów zawierają również wszystkie style dla rdzenia (style wspólne dla wszystkich komponentów), więc musisz dołączyć tylko jeden plik CSS dla materiału Angular w swojej aplikacji. Możesz dołączyć plik motywu bezpośrednio do aplikacji z angular/material/prebuilt-themes. Oto dostępne gotowe motywy:

- ciemnofioletowy-bursztynowy.css

- indygo-różowy.css

- różowo-niebieskoszary.css

- fioletowo-zielony.css

Używasz tutaj Angular CLI, więc możesz po prostu dołączyć żądany styl do globalnego pliku src/styles.css. Pierwotnie ma ten wstępny kod wstępny:

html, body { height: 100%; }

body { margin: 0; font-family: 'Roboto', sans-serif; }

Dodaj następującą instrukcję importu na górze dokumentu: @import "~@angular/material/prebuilt-themes/indigo-pink.css"; Gdy masz otwarty plik src/style.css, możesz również utworzyć styl dla kontenera, akapitu i przycisku, których możesz używać w swojej aplikacji na swoich stronach.

p {

padding-left: 20px;

```
font-size: 12px;
}
.container {
margin-right: auto;
margin-left: auto;
padding: 20px 15px 30px;
width: 750px;
}
button {
color: #ffffff;
background-color: #611BBD;
border-color: #130269;
display: inline-block;
margin-bottom: 0;
font-weight: normal;
text-align: center;touch-action: manipulation;
cursor: pointer;
white-space: nowrap;
padding: 6px 12px;
font-size: 12px;
line-height: 1.42857143;
border-radius: 4px;
-webkit-user-select: none;
-moz-user-select: none;
-ms-user-select: none;
user-select: none;
}
vertical-align: middle;
```

Tworzenie treści

W tym momencie masz szkielet aplikacji z nagłówkiem, treścią i stopką. Treść można przełączać między stroną początkową a stroną transferu, zmieniając adres URL w przeglądarce. Zaimportowałeś również i wstrzykiłeś moduły Material oraz skonfigurowałeś globalne style dla swojej aplikacji. Następnym

krokiem jest utworzenie rzeczywistej treści, która zastąpi tymczasową wiadomość tekstową, którą umieściłeś w komponentach nagłówka, stopki i startu.

Komponent stopki

W przypadku elementu stopki po prostu zamienisz wiadomość dotyczącą praw autorskich Twojej firmy. Aby to zrobić, wystarczy otworzyć src/app /components/footer/footer.component.html i zastąpić domyślny kod

< p >

footer works!

< /p >

Zastąp kod, tworząc kontener div z dodanym stylem do globalnego pliku CSS.

< div class="ng-scope" >

< div class="container" >

< p >Copyright (c) 2019 Company Name. All Rights Reserved.< /p >

< /div >

< /div >

Zamierzasz również stworzyć określony styl dla komponentu stopki, więc za każdym razem, gdy użyjesz tagu p, twoja czcionka będzie miała rozmiar 12 pikseli bez dopełnienia po lewej. Otwórz src/app/components/footer/footer.component.css i wstawić, co następuje:

p {

padding-left: 0;

font-size: 11px;

}

Zauważ, że znacznik <p> został zdefiniowany dwukrotnie, raz w globalnym pliku CSS i raz na poziomie komponentu. To, co się wydarzy, to to, że globalny tag < p > zostanie nadpisany przez komponent <p>, więc możesz użyć tagu <p> dla swojej stopki i innego tagu < p > dla innych komponentów, takich jak start i transfer strony, zachowując kod HTML wolny od kodu CSS.

Komponent nagłówka

W przypadku komponentu nagłówka utworzysz menu nawigacyjne, aby móc przełączać się między stroną początkową a stroną transferu. Aby uzyskać style specyficzne dla komponentu nagłówka, otwórz src/app/components/header/header. component.css i dodaj style listy nawigacyjnej.

.nav {

margin-bottom: 0;

padding-left: 0;

list-style: none;

}

```
li {
display: block;
float: left;
width: 100px;
height: 25px;
padding: 5px;
}
.nav>li>a {
margin-bottom: 0;
padding-left: 0;
font-weight: 500;
font-size: 12px;
text-transform: uppercase;
position: relative;
}
```

Dla src/app/components/header/header.component.html tworzysz kontener i listę dwóch linków do stron startowych i transferowych. W tym celu zastąp początkowy kod:

```
<p>
header works!
</p>
```

with the following;

```
<div class="ng-scope">
<div class="container">
<ul class="nav">
<li>
<a routerLink='/start'>home</a>
</li>
<li>
<a routerLink='/transfer'>transfer</a>
</li>
</ul>
```

Działający dapp zawiera teraz podstawową stylizację i funkcjonalną nawigację.

Przenieś komponent

Komponent transferu będzie zawierał formularz, który prześlesz, aby przelać monety z jednego adresu konta na inny. Będziesz korzystać z modułu formularzy, aby przyspieszyć tworzenie formularza. Aby to zrobić, musisz uwzględnić moduły formularzy Material FormsModule i ReactiveFormsModule w app.module.ts, tak jak w przypadku innych modułów Material. Otwórz src/app/app.module.ts i dodaj następującą instrukcję importu:

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

Chcesz również zaktualizować instrukcję importu.

imports: [

FormsModule,

ReactiveFormsModule,

..

]

Będziesz używać znacznika <mat-form-field>, który reprezentuje komponent, który otacza kilka komponentów Angular Material i stosuje typowe style pól tekstowych, takie jak podkreślenie, pływająca etykieta i komunikaty podpowiedzi. Przyspieszy to rozwój, ponieważ nie będziesz musiał ich wszystkich implementować i testować na wielu urządzeniach/przeglądarkach. Pole formularza to opakowujący komponent o nazwie <mat-form-field>. Możesz użyć dowolnych kontrolek pól formularza (takich jak input, textarea, list itp.). Informacje o formach mat znajdziesz tutaj: https://material.angular.io/components/form-field/overview. W przypadku src/app/components/transferztransfer.component.ts zaktualizujesz początkowy kod. Najpierw musisz zaimportować komponenty, których będziesz używać; w takim przypadku musisz zainicjować klasę i użyć formularza, kontroli formularza i walidatorów.

import {FormBuilder, FormControl, FormGroup, Validators} from

'@angular/forms

Następnie musisz zaktualizować definicję komponentu, aby zaimplementować metodę OnInit.

export class TransferComponent implements OnInit {

Będziesz używać flagi, aby wskazać, czy formularz został przesłany i utworzyć instancję grupy formularzy, a także obiekt o nazwie user, w celu przechowywania informacji o użytkowniku.

formSubmitted: Boolean = false;

userForm: FormGroup;

user: any;

Aby zatwierdzić formularz, zdefiniujesz wiadomości w przypadku gdy formularza nie jest poprawnie wypełniony. Każda kontrolka formularza musi być zdefiniowana z wymaganymi polami i komunikatami.

```
account_validation_messages = {

'transferAddress': [

{ type: 'required', message: 'Transfer Address is required' },

{ type: 'minLength', message: 'Transfer Address must be

42 characters long' },

{ type: 'maxLength', message: 'Transfer Address must be

42 characters long' }

],

'amount': [

{ type: 'required', message: 'Amount is required' },

{ type: 'pattern', message: 'Amount must be a positive

number' }

],

'remarks': [

{ type: 'required', message: 'Remarks are required' }

]

};
```

Podczas tworzenia konstruktora musisz dołączyć składnik FormBuilder, aby móc wygenerować formularz.

```
constructor(private fb: FormBuilder) { }
```

Kiedy twój komponent otrzyma init, ustawisz flagę formSubmitted na false i ustawisz domyślne wartości dla informacji użytkownika. Następnie wywołasz metodę pobierania konta użytkownika i salda, którą zaimplementujesz później. Na koniec wywołasz metodę createForms, która wygeneruje formularz.

```
ngOnInit() {

this.formSubmitted = false;

this.user = { address: '', transferAddress: '', balance: '',

amount: '', remarks: ''};

this.getAccountAndBalance();

this.createForms();
```

```
}
```

The createForms method will generate the form controls by passing

the validators and data.

```
createForms() {
this.userForm = this.fb.group({
transferAddress: new FormControl(this.user.transferAddress,
Validators.compose([
Validators.required,
Validators.minLength(42),
Validators.maxLength(42)
])),
amount: new FormControl(this.user.amount, Validators.
compose([
Validators.required,
Validators.pattern('^[+]?([.]\\d+|\\d+[.]?\\d*)$')
])),
remarks: new FormControl(this.user.remarks, Validators.
compose([
Validators.required
]))
});
}
```

Metoda getAccountAndBalance ustawi adres i saldo konta użytkownika; na razie używasz fikcyjnych danych, ale rzeczywistą usługę zaimplementujesz w dalszej części tej części.

```
getAccountAndBalance = () => {
const that = this;
that.user.address = '0xd8d0101f83e79fb4e8d21134f5325e64816b
d6a0';
that.user.balance = 0;
// TODO: fetch data
}
```

Wreszcie, gdy prześlesz formularz, potrzebujesz metody do obsługi danych i zadzwoń do serwisu. submitForm zostanie użyty przez sprawdzenie, czy formularz jest poprawny, a następnie wywołasz składnik usługi, który utworzysz.

```
submitForm() {

if (this.userForm.invalid) {

alert('transfer.components :: submitForm :: Form invalid');

return;

} else {

console.log('transfer.components :: submitForm :: this.

userForm.value');

console.log(this.userForm.value);

// TODO: service call

}

}

}
```

W przypadku transfer.component.html ustawisz tag formularza, aby wywoływał metodę submitForm po przesłaniu formularza.

```
<form [formGroup]="userForm" (ngSubmit)="submitForm()" novalidate autocomplete="off">
```

Następnie utworzysz wrapping div i użyjesz data binding, aby wyświetlić adres i saldo konta użytkownika.

```
<div class="container">

<div class="transfer-container">

<div>

Address: {{user.address}} <br/>

Balance: {{user.balance}} Eth

</div>
```

Zauważ, że użyłeś stylu transfer-container, którego jeszcze nie zdefiniowałeś; zdefiniujesz go w pliku CSS i będzie on używany do sformatowania formularza. Do kontroli formularzy potrzebne są pola do wprowadzania danych dla konta, na które przelewasz środki, kwota i wiadomość. Musisz także skonfigurować swoje walidacje.

```
<mat-form-field>

< input matInput placeholder="Transfer Address"

name="transferAddress" formControlName="transferAddress"

maxlength="42" minlength="42" required>
```

```
< mat-error *ngFor="let validation of account_
validation_messages.transferAddress">
<mat-error *ngIf="userForm.get('transferAddress').
hasError(validation.type) && (userForm.
get('transferAddress').dirty || userForm.
get('transferAddress').touched)">{{validation.
message}}</mat-error>
</mat-error>
</mat-form-field>
<mat-form-field>
< input matInput placeholder="Amount" name="amount"
formControlName="amount" required>
< mat-error *ngFor="let validation of account_
validation_messages.amount">
<mat-error *ngIf="userForm.get('amount').
hasError(validation.type) && (userForm.
get('amount').dirty || userForm.get('amount').
touched)">{{validation.message}}</mat-error>
</mat-error>
</mat-form-field>
<mat-form-field>
< input matInput placeholder="Remarks" name="remarks"
formControlName="remarks"
maxlength="42" required>
< mat-error *ngFor="let validation of account_
validation_messages.remarks">
< mat-error *ngIf="userForm.get('remarks').
hasError(validation.type) && (userForm.
get('remarks').dirty || userForm.get('remarks').
touched)">{{validation.message}}</mat-error>
</mat-error>
```

</mat-form-field>

Na koniec pamiętaj, aby zamknąć elementy div i formularz, a także dołączyć przycisk przesyłania.

<button type="submit">Transfer Ether</button>

</form>

W przypadku transfer.component.css będziesz używać div transfer-container do sformatowania formularza w poziomie.

.transfer-container {

display: flex;

flex-direction: column;

}

.transfer-container > * {

width: 100%;

}

Otóż to. Teraz możesz sprawdzić swój dapp w przeglądarce i powinieneś być w stanie zobaczyć domyślne dane użytkownika, przetestować formularz, zweryfikować go i przesłać formularz.

Dyrektywy Angular

Tworzenie dyrektyw w Angular daje możliwość tworzenia własnych niestandardowych znaczników HTML za pomocą zaledwie kilku linijek kodu, tak jak widzieliśmy w formularzu Materiał. Mogłeś dołączyć niestandardowe tagi, które otaczają wiele komponentów. Na wysokim poziomie dyrektywy są znacznikami na elemencie DOM. Te znaczniki mogą wskazywać na dowolny komponent DOM, od atrybutu po nazwę elementu, a nawet komentarz lub klasę CSS. Te znaczniki następnie informują kompilator HTML AngularJS o dołączeniu określonego zachowania lub przekształceniu całego elementu DOM i jego dzieci w oparciu o określoną logikę. Angular ma wiele wbudowanych dyrektyw. Jednak podczas rozwoju jest duża szansa, że będziesz tworzyć własne dyrektywy. Twój dapp jest teraz prosty, więc nie musisz tworzyć żadnej dyrektywy, a wyjaśnienie tego wykracza poza zakres tego rozdziału. Kiedy musisz wygenerwać dyrektywę szkieletową, użyj Angular CLI tak, jak generowałeś inne komponenty.

> ng generuj dyrektywę {nazwa-dyrektywy}

Chociaż nie tworzysz dyrektywy w swojej aplikacji, chciałem przedstawić Ci tę koncepcję, ponieważ jest to integralna część tworzenia projektu Angular.

Streszczenie

W tej części zagłębiłeś się w to, czym jest dapp i przyjrzałeś się klasyfikacji i projektom dappa. Nauczyłeś się, jak rozpocząć własny projekt dapp, dzieląc proces na pięć kroków: napisanie białej księgi, uruchomienie ICO, opracowanie dappa, uruchomienie go i marketing swojego dappa. Następnie zastanawiałeś się, dlaczego warto korzystać z Angulara. Następnie utworzyłeś aplikację Angular, najpierw upewniając się, że zainstalowano wymagania wstępne i instalując interfejs CLI Angular. Następnie stworzyłeś projekt Angular i obsłużyłeś aplikację. Następnie nauczyłeś się importować swój projekt Angular do WebStorm lub tworzyć nowy projekt. Przyjrzałeś się elementom tworzącym Angular, takim jak komponenty, moduły i dyrektywy. Nauczyłeś się również, jak stylizować dapp, rozumiejąc architekturę w stylu Angular i pracując z materiałem Angular Material. Zacząłeś budować komponenty i tworzyć treści; podzieliłeś swoją aplikację na stopkę, nagłówek i treść oraz utworzyłeś niestandardowy składnik o nazwie transfer, który zawiera formularz, aby móc później przesyłać tokeny. W następnej części utworzysz inteligentną umowę transferu i projekt rozwoju Truffle, a także połączysz się z siecią deweloperską Ganache. Dowiesz się, jak pracować z siecią Ethereum za pośrednictwem Truffle i przetestujesz swój inteligentny kontrakt. Połączysz również swój dapp z biblioteką web3 sieci Ethereum Network i połączysz się przez MetaMask.