1. **WEBPACK**

Webapck bundluje pliki w jeden wielki plik (wyjątek: biblioteki 3rd parties sa umieszczone w osobnym wilkim pilku) poprzez tworzenie z plikow js/css/html modulow.

Konfiguracja jest zrobiona pod dev i prod.

HMR potrafi przebudowac tylko zmienione moduly bez koniecznosci budowania calego pliku od nowa (z pomoca eval-source-maps).

WebpackMiddleWare integruje pliki do serwera (zadnych plikow na dysku, tylko pamiec, bez refreshowania strony).

W produkcji po kliencie moduly css i html i pliki media sa wstrzyknniete do plikow, w devie sa wstrzykniete w „style” albo js (wyjatek,gdy url nie może wstrzyknac zdjecia w format base 64 i paleczke przejmuje file loader

1. **OBSERVABLE SOCKET**

Jest to moduł wrapujący metody socket.io które zostaną uzyte po stronie serwera i klienta.

Po serwerze instancje socketa clienta zostana uzyte do rejestrowania klienta w poszczegolnych API modulu

Po kliencie zostanie uzyty socket serwera do którego będą emitowane eventy wraz danymi z UI

1. **registerCallback**

jest to w rodzaju rejestrator action/eventow jakie wysle dany klient. Sa rejestrowane raz, zostaja utworzone listenry pod te eventy i utwrzony subject do którego klient subsrybuje. Po zarejestrowaniu nie musi ponownie ustawiac listenerow co przeklada się na O(1) efektywnosc aczkolwiek pamiec rośnie

1. **onAction (Server side)**

Zostanie nasluchany event z danymi od klienta. Serwer otrzyma dane i sprobuje wykonac blok kodu w try. Najpierw wykona przekazany callback. Jeżeli zwrotna wartosc cb to nie observable to wyemituje wartosc, jezeli to observable to subsrybujemy do niego i generujemy do komponenta/listnera który subsrybuje wartosci

1. **Modular Server**

Poszczegolne żądania do serwera, zostana obsluzone przez poszczegolne moduły ( 3 moduly). Głowny serwer wywola ich poszczegolne init funkcje, a nastepnie zarejstruje w modulach poprzez registerClient i da znak ze jest zarejestrowany we wszystkich modulach

Client.onActions zarejestruje listenery na emitowane przez klienta eventy.

1. **Klasa Komponent**

BaseComponent sluzy do inicjowania zmiennych takich jak tablica dzieci, handlerow na mount i demounta. ElementComponent sluzy do utworzenia elementu ($element) i dolaczeniu go do shella ($mount). Komponenty odpowiadaja tylko za mount nodea i elementy swoje podrzędne, żadnych side effektów ani manipulacji poza nimi.

1. **Klasa ModuleBase**

Klasa dla modulow inicujacych gdy klient zażąda od web servera danych i inicjuje swoje API

1. **STORE**

Services będą odpowiadaly za przekazywanie nowego stanu do UI oraz modelu danych. Store będzie immutable i zostanie przekazany nowy stan porownujac ze starym czy zostal zmieniony.

bodajze zainicjuja listenery na stronei web serwera i zmapuja do eventa funkcje która wykona operacje na stanie.

bedzie wywolywal view-model koncepcje i dispachowal eventy?

**USER MODULE**

Modul ten jest zainicjowany (init funkcji bazowej) przy startowaniu serwera. Jego API zostaje udostępnione w momencie połączenia klienta z serwerem. Serwer wywołuje funkcje ‘registerClient’ i przekazuje do niej instancje klienta (połączenie socketa).

‘registerClient’ rejestruje i tworzy listenery dla poszczegolnych akcji/eventów (onActions) wyemitowanych przez klienta, a szczególniej przez uslugę UserStore. Komponent tylko subsrybuje przez usługę i uzyskuje dane wraz typem akcji jaką serwer wykonał

Zwraca listę zalogowanych użytkowników oraz dane gdy nowy użytkownik zaloguje się, poprzez wyemitowanie ‘user:list’ przez serwera akcji zwrotnej ‘connect’ do klienta.

Na poczatku wysle pusta liste, ponieważ inicjujaca operacja to zwrocenie listy zalogowanych uzytkownikow

Druga operacja to zalogowanie uzytkownika do serwera i zwrocenie klientu jego danych autoryzacyjnych

Logowanie jest poprzez metode usługi ‘login’ (wyemitowanie ‘auth:login’) , gdzie zostanie wywolny po stronie modułu loginClient$, ponieważ moduł ten zarejestrował tą akcję i listenery.

*ASIDE: validator zwraca pusty obiekt jeżeli nie ma bledow, jeżeli sa to zwraca wiadomosci z poszczególnych errorów*

W główniej mierze walidacja to dodanie właściwości ‘auth’ do instancji klienta, gdzie zostaną dodane dane autoryzacyjne, albo sprawdza się czy obiekt istnieje i wysłanie wiadomości do klienta, że prawdopodobnie username jest wzięty. Następnie dane auth i event ‘users:added’ zostaną wyemitowane

User Store przechwyci i utworzy z tego observable i wykona operacje ze zmiana starego stanu na nowy poprzez funkcje zmieniajace stan i zwracaja nowe. Te funkcje mają dane w parametrze i zwracją funkcję która zostanie wywołana przy reducerze rxjs

**Niewiadome: w jakim momencie dane są przekazane do streamy, aby funkcja mapująca posiadała je w argumencie?**