

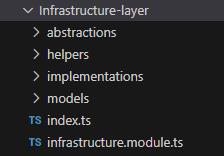
die rot durchgestrichenen kannst ignorieren

die mit pfeil daneben sind die wichtigen

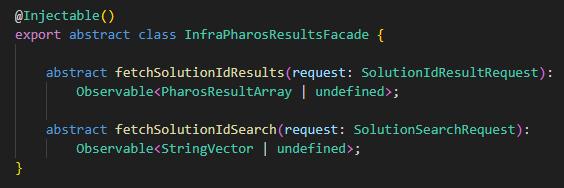
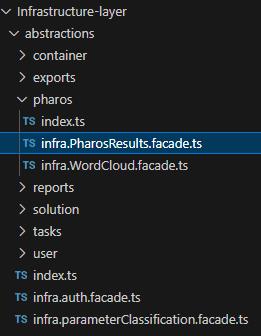
wir haben auch noch module...das ist noch etwas altmodisch in Relation zu Angular 20

Infrastructure Layer

beinhaltet nur Services, die irgendwelche API-Calls ausführen und selbst keinen State besitzen, also nur Methoden, die etwas zurückgeben, aber nichts in der Klasse selbst speichern. Die Ordnerstruktur innerhalb eines Layers sieht im Application-Layer genauso aus.

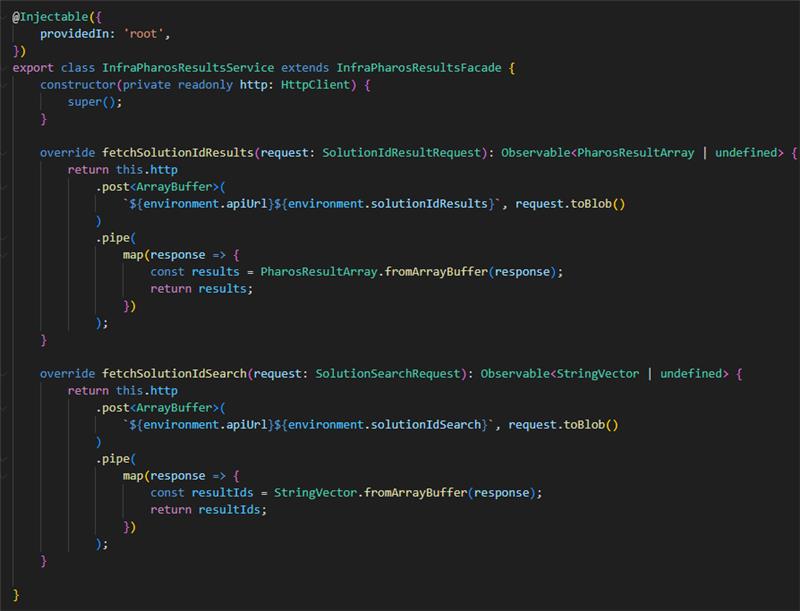
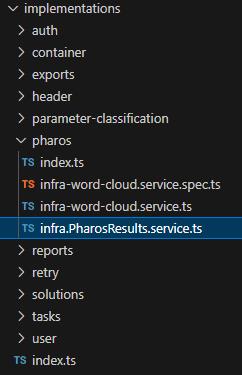


abstractions -> da sind die ganzen abstrakten klassen drinnen, also die definition was welcher service macht, hier ein beispiel:



helpers -> beinhaltet nur hilfsklassen

implementations -> sind dann die tatsächlichen implementierungen zu den abstrakten klassen, hier das gleiche beispiel von vorher aber implementiert:

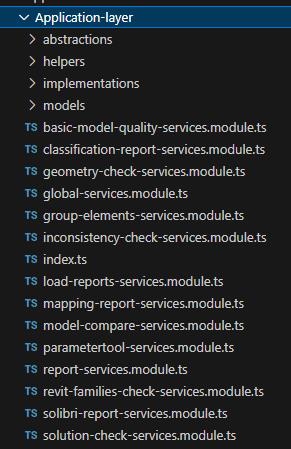


models -> beinhaltet zusätzliche modelle...hier könntest du dir die definitionen für die jsons schreiben, die vom backend kommen

ACHTUNG!: Die gezeigte Ordnerstruktur hat sich als etwas mühsam erwiesen, weil man dann immer zwischen implementations und abstractions hin und her wechselt und das geht einem auf Dauer richtig 'am zager'! Meine Empfehlung: Mach dir direkt unterhalb vom Abstraction und Infrastructure Layer die Ordner die du für unterschiedliche Aufgaben brauchst. In unserem Fall wäre es, um auf das gezeigte Beispiel zurückzukehren, der Ordner 'pharos'. Der wäre besser direkt unter Infrastructure-layer anzusiedeln und in dem kannst dann direkt die abstrakten Klassen und Implementierungen ablegen.

Application Layer

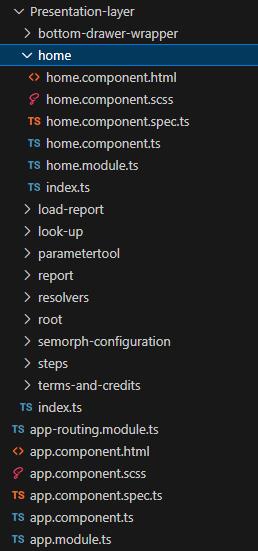
enthält auch nur Services, die die Programmlogik verwalten und im Unterschied zum Infrastructure Layer sehr wohl auch Zustände haben (das könnten jetzt Signals sein oder BehaviorSubjects von rxjs). Der Application Layer ruft Funktionen aus dem Infrastructure Layer auf und reicht die errechneten/transformierten Ergebnisse weiter an den Presentation Layer. Von der Ordnerstruktur ist der genauso aufgebaut wie der Presentation Layer:



Gleiches Konzept mit gleicher Warnung wie zuvor. Mach es lieber anders, weil so ist es nur mühsam.

Presentation Layer

Enthält die ganzen Komponenten wie bei klassischem Angular. Diese injecten die Services aus dem Application Layer und stellen die Dinge einfach nur dar und schicken aber auch Outputs an den Application Layer. Die Komponenten verwenden ihrerseits Komponenten aus unserer UI-library. Das heißt der Presentation Layer ist in dem Fall Schnittstelle zwischen Application Layer und UI-Komponenten (die können jetzt entweder aus einer Library kommen oder eigens in einem Ordner angelegt werden)



Wie du siehst für jede Page in der Web-Applikation gibt es einen Ordner und darin Komponenten oder wiederum Ordner, die Komponenten beinhalten wie zum Bsp:

