BEWERBERAUFGABE – LUNCH TIME LÖSUNG.

# AUSBAUSTUFE 4 (CONSULTANT).

Es gibt folgende Methode, um die Daten vom Server zu aktualisieren:

* Basic intervall based refresh
* Event driven refresh
* RxJs
* Web sockets

# Basic intervall based refresh.

Der Beispielcode könnte folgendermaßen aussehen:

export class OrderListComponent implements OnInit, OnDestroy {

  private interval?: number;

  ngOnInit(): void {

      this.interval = window.setInterval( () => {

              this.orderService.retrieveOrders().subscribe(

                response => {

                  this.orders = response;

                })

      }, 5000);

  }

   ngOnDestroy(): void {

    if(this.interval) {

      clearInterval(this.interval);

    }

}

## Event driven refresh

„Event-driven Refresh“ ähnelt dem „Basic Interval-based Refresh“. Die periodische Datenabfrage erfolgt in einem Angular Service. Wenn sich die Daten ändern, erzeugt der Service ein Observable-Event. Die Komponente sollte dieses Event abonnieren und bei dessen Eintreten die Daten neu rendern.

## rxJs.

RxJS bietet maximale Kontrolle darüber, wann Daten aktualisiert werden. Der Code könnte folgendermaßen aussehen:

rxJs bietet maximal Kontrol, wenn data aktualisiert wird. Der Kode kann so aussehen:

export class OrderListComponent implements OnInit, OnDestroy {

  ngOnInit(): void {

interval(1000)

.pipe(

switchMap( () => this.orderService.retrieveOrders())

)

.subscribe((data) => {

                  this.orders = data;

}

  }}

## Web sockets

Die oben beschriebene Methode verwendet eine Pull-Abfrage. Als Alternative dazu nutzt die WebSocket-Methode einen Push-Mechanismus: Der Server sendet eine Nachricht, und der Angular-Code wird sofort ausgeführt. WebSockets eignen sich besonders gut für Live-Daten, da sie ein Rendering in Echtzeit ermöglichen. Der Code könnte folgendermaßen aussehen:

export class OrderListComponent implements OnInit, OnDestroy {

  private myWebSockt: WebSocketSubject<any> = new WebSocketSubject<any>(‘ws:localhost:8080’);

  ngOnInit(): void {

      this.myWebSocket.subscribe((data) => {

                  this.orders = data;

  }

}