Java Systemutveckling

Java Fundamentals

RESTORY...

Agenda

- Hur ska man börja?
- Project Scope.
- Getting your hands dirty.
  - Wireframing
  - Models & Relationships
  - FlowCharts
- Projekt

RESTORY...

Planering & utveckling av applikationer

Att planera en full-stack applikation

RESTORY...

Hur ska man börja?

I nästa del av denna presentation kommer vi att titta närmare på den viktigaste delen av "Getting started" -ritualen när vi ska planera en full-stack applikation.

Det finns absolut inget behov av att besvära huvudet med exakt hur man kommer att börja implementera affärslogiken i kod eller vilket ramverk man kommer att använda för att skapa en API.

I vårt fall vet vi redan att vi kommer att använda Java, men det är inte viktigt just nu. Det som är viktigt är projektets *scope*.

Detta är inte en *tutorial*, det här är helt enkelt en äventyrlig reflektion över hur man kan arbeta och hitta rätt *approach*. Ingenting är skrivet i sten men att följa några enkla steg kan hjälpa i processen.

> *"Getting started is the hardest part."*
>
> William Shakespeare*
>
> *(not really)

RESTORY...

Projektets Scope

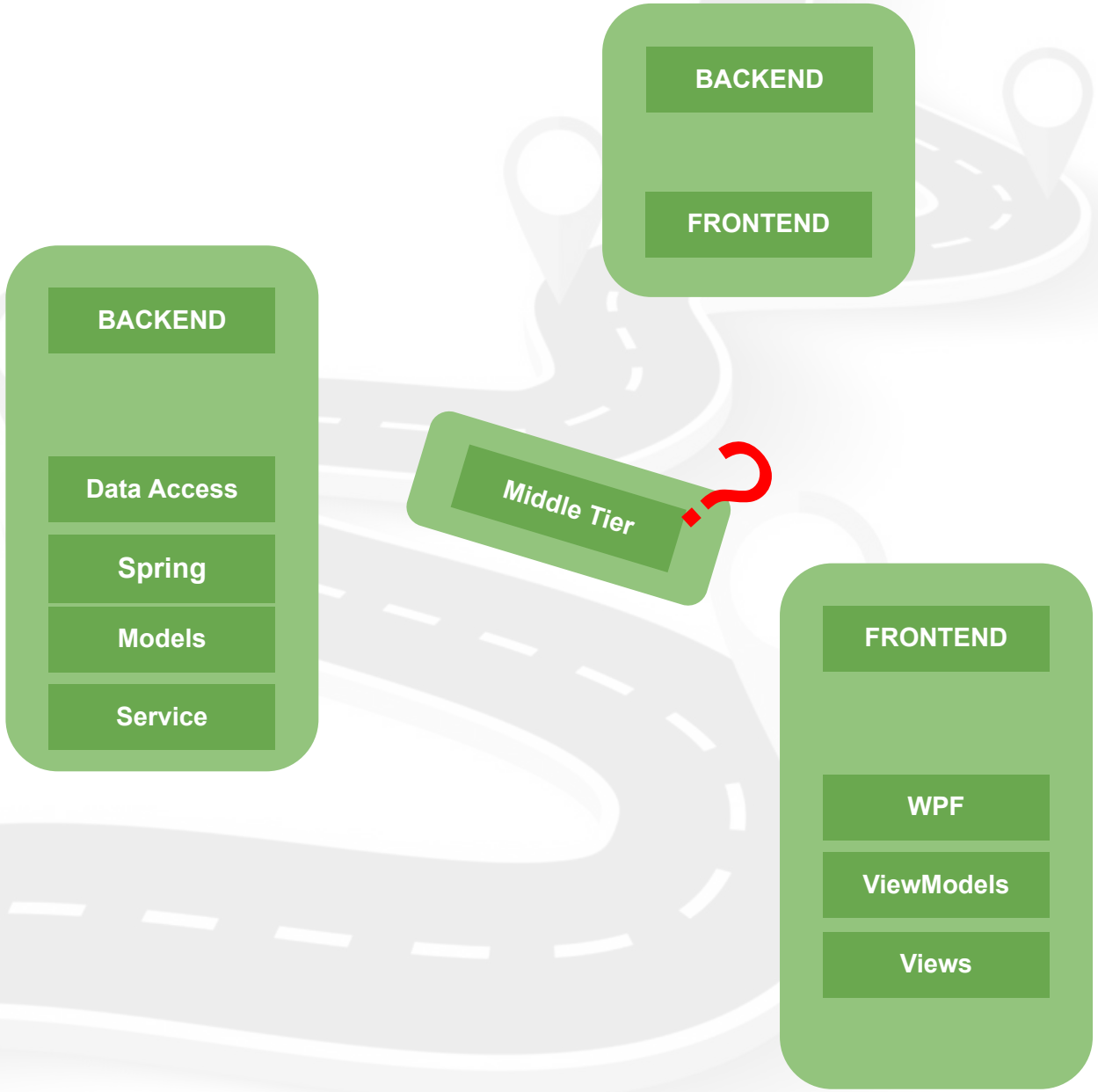
Vad menar vi med "Project Scope"?

Att definiera omfattningen (*scope*) av ett projekt är av största vikt i strävan efter "clean" mjukvaruutveckling.

Vi har naturligtvis hört talas om SOLID -principen men det är inte det vi siktar på i det här fallet. Vårt mål är helt enkelt att definiera vårt "CASE" steg för steg med all tillgänglig information som finns i projektspecifikationerna.

Vårt ***Project Scope*** måste vara strikt *objective-oriented* med mer eller mindre tydliga beskrivningar av hur utvecklingen av en specifik del av applikationen kommer att bemötas. Starten ska vara så generell som möjligt och samtidigt gå ner till de specifika detaljerna för en specifik implementering på ett sådant sätt att vi inte börjar beskriva utvecklingen genom att använda en specifik teknologi som utgångspunkt.

Vi kan naturligtvis börja skapa en lista över Stack teknologierna som kan användas för den aktuella uppgiften.

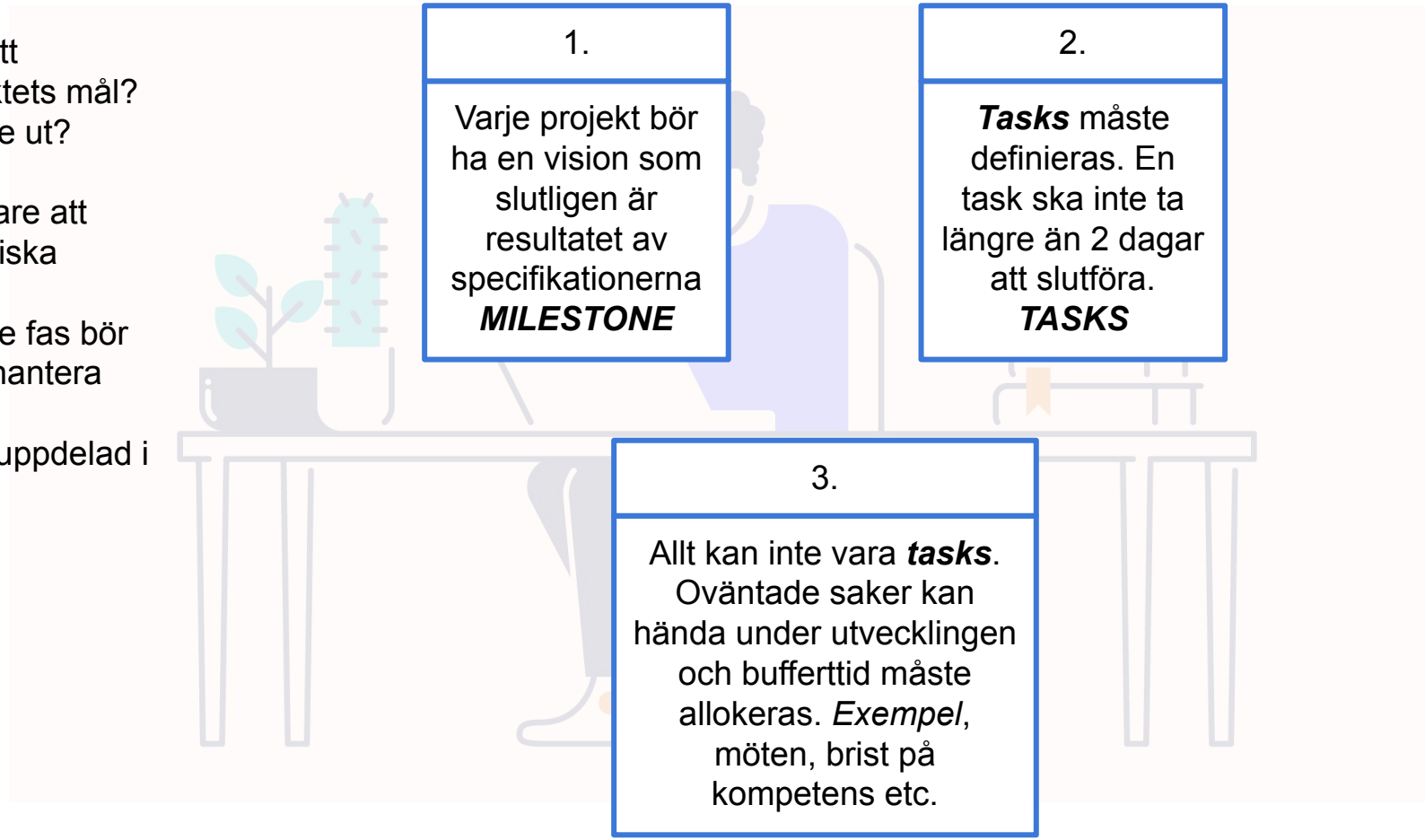


RESTORY...

Projektets Scope

Att definiera en Project Scope handlar också om att dokumentera en lista specifikationer. Vad är projektets mål? Vad ska levereras? Hur ska projektets *deadlines* se ut?

- Att definiera ett mål till projektet gör det enklare att dela upp arbetet i *milestones* som har realistiska *deadlines*.
- Huvudidén är att ha kontroll över den tid varje fas bör ta att utvecklas och på så sätt också kunna hantera motgångar och bristande kompetens.
- Enligt den Agila Metoden är arbetets *Scope* uppdelad i så kallade "*Items of Development*" (läs mer).



RESTORY...

Wireframing

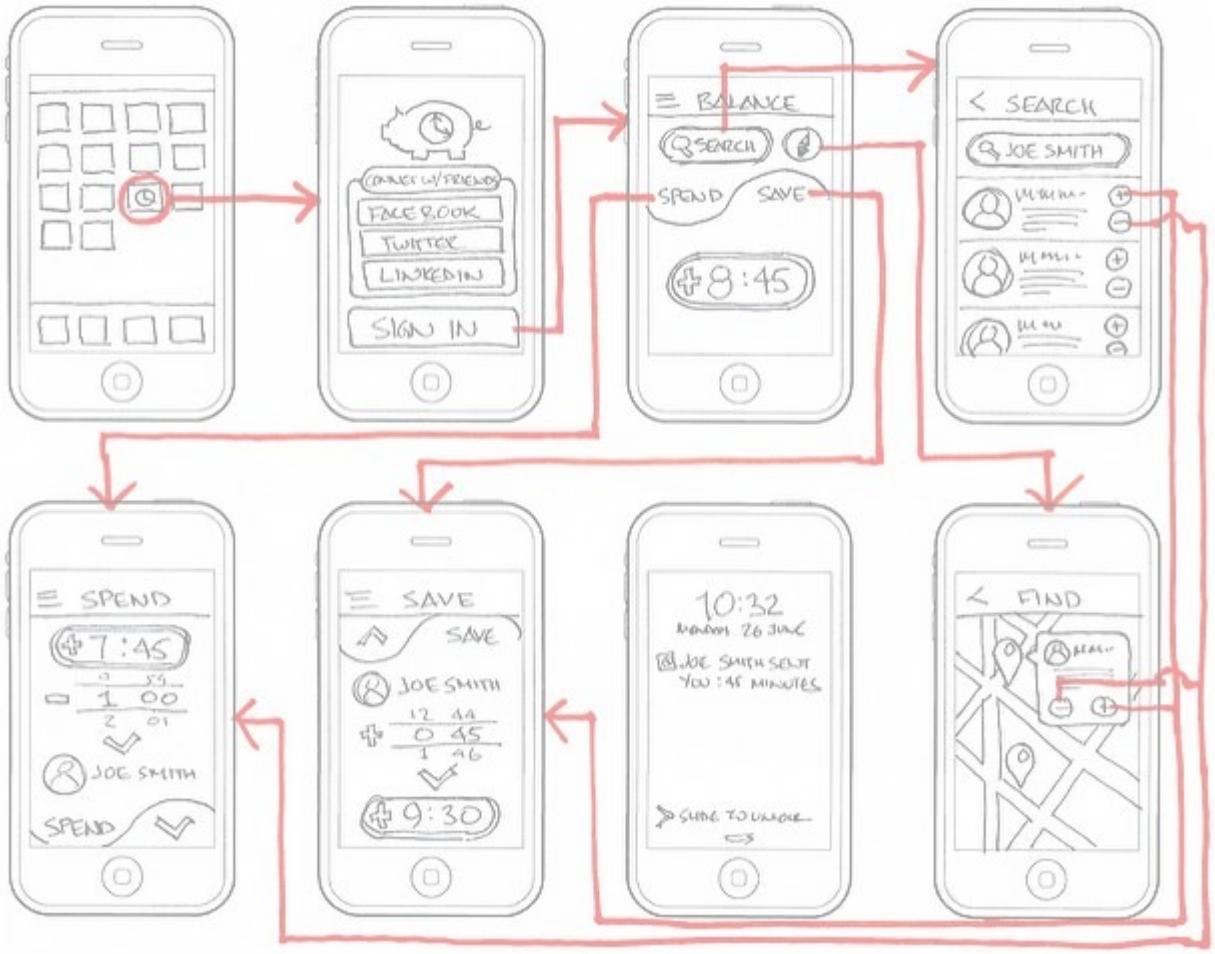
Wireframing -processen är vanligtvis fokuserad ur funktionalitetens synvinkel, d.v.s. användarberättelserna.

Ex. Ska vi redogöra för autentisering och auktorisering? Vår applikation borde kanske innehålla en vy som gör att användaren kan logga in eller skapa ett konto. Dessutom måste användaren vara auktoriserad i vyer som endast är avsedda för en användare som redan har autentiserats.

Wireframing är bland annat konsten att skapa en användarupplevelse under vissa förutsättningar baserat på användarberättelserna. Vi kommer att berätta för användaren exakt vad han ska kunna göra i applikationen.

Att tänka på:

- Fokusera på funktionalitet
- Skapa en hierarki av statiska och dynamiska element
- Håll *Wireframes* enkla och fokuserade på uppgiften
- Skapa små block av användarinteraktioner med en definierad sekvens.
- *Wireframes* kan vidareutvecklas, vi måste också ta hänsyn till detta.



RESTORY...

Planera Backend

Om vi tänker i Backend -*terms* kanske vi vill skapa en API-tjänst med väldefinierade endpoints som kommer att avslöja resurser. I vårt fall vet vi redan att vi kommer att använda Java, någon slags ORM som Hibernate, MySql, kanske också MVC-design mönstret.

Alla applikationer kommer inte att fungera på samma sätt. Om vi vet att vår applikation kommer att användas av många som skulle vilja ha koll på deras interaktioner med din applikation bör vi förmodligen tänka på autentisering och auktorisering.

Om vi tar till exempel en **Movie Recommendations App** där en användare kan söka efter en film beroende på variabler som *genre*, *rating* etc. Vill vi kanske att användaren ska kunna göra detta utan att behöva vara inloggad? Hur skulle vi då kunna tackla en *feature* i vår applikation som ger användaren förmågan att lägga till en **WatchList** eller **FavouriteList** denne själv skapat?

Kan en användare som inte autentiserats auktoriseras att kunna kommentera under en **CommentSection** view?

Det är nödvändigt att ta lite tid och skissa hur processen för autentisering och auktorisering kommer att fungera

RESTORY...

Planera Backend

Vi kommer sannolikt att använda JSON för att kommunicera med vår frontend men det är viktigt att vi bestämmer exakt hur *Requests* och *Responses* kommer att se ut. (LÄS HÄR)

Här börjar vi komma till några API -designfrågor. Ta ett papper och skriv ner alla potentiella "*objects*" i din backend. Om du till exempel byggde en **Car Reservation App** kan det här vara saker som "användare", "bil", "bokning", "faktura", "kontrakt", etc. Tänk igenom dem alla och ta reda på hur du gör vill ha dem representerade i ditt API. Du vill tänka på hur svaret för var och en kommer att se ut, eftersom det bör vara konsekvent i hela ditt API.


```
[
    "user": {
        "id": 5,
        "email": "alfia@gmail.com",
        "is_admin": false,
        "manager_id": null,
        "created_at": "2017-07-29T13:52:15.683Z",
        "updated_at": "2017-07-29T13:52:15.683Z",
        "authentication_token": "oA98XDX5aMsie1Lh7BWP",
        "organization_id": 3
    },
    "blocks": [
        {
            "id": 5,
            "name": "C1"
        },
        {
            "id": 6,
            "name": "C2"
        }
    ]
}
```

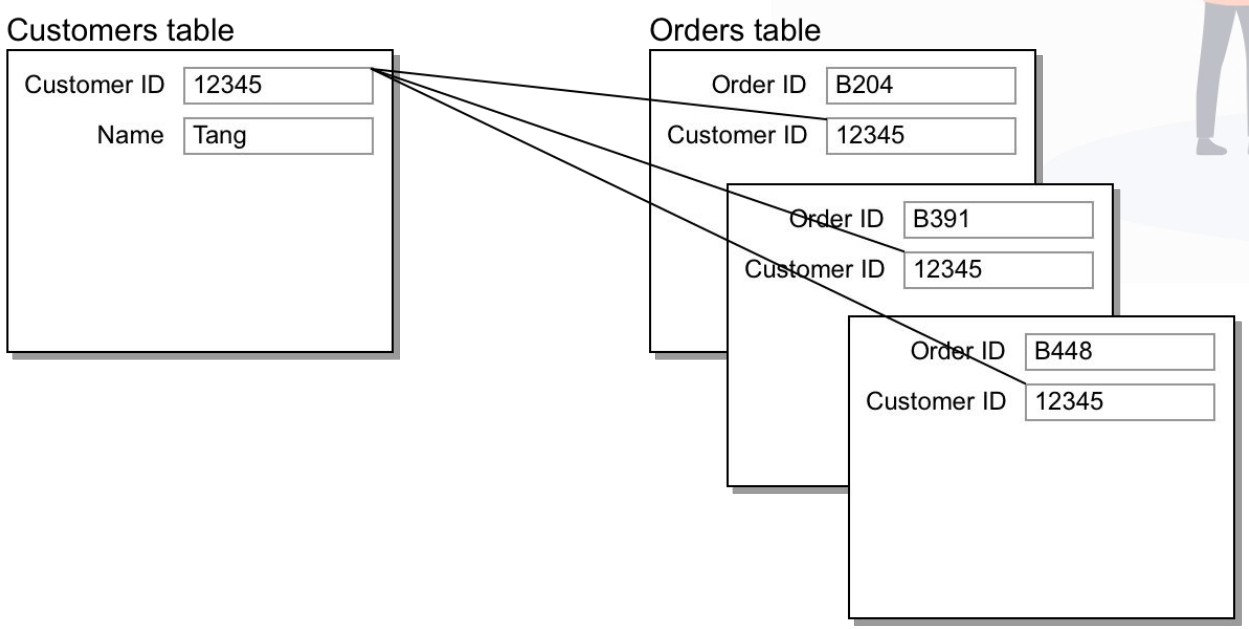

RESTORY...

Planera Backend

När vi har bestämt vilka objekt vår applikation behöver arbeta med kommer interaktionsdelen.

Hur kommer ett objekt att interagera med ett annat? Här måste vi tänka på relationer så att vår databas är korrekt strukturerad.

Ett exempel kan vara förhållandet en specifik användare i en **Movie Recommendations App** kommer att ha med en modell av *type* **MovieList** där användaren kan skapa, uppdatera, radera en eller flera av dessa listor som vanligtvis kommer att struktureras relationellt som *one-to-many*.



RESTORY...

Planera Backend

Schema är det som avgör vilka attribut du vill att dina modeller ska ha och spara i databasen.
Om man till exempel har en användare kan man välja att ge denne ett *name*, *lastname* och *email*. Man måste tänka på vilka attribut varje modell behöver för att kunna ge applikationen mer flexibilitet. Man kan dock fortfarande lägga till fler attribut senare om man tror att en av modellerna behöver ha mer information.

```
class Store {
    private UUID id;
    private String name;
    private String country;
    private String city;
    private List<Product> products;
}
class Product {
    private UUID id;
    private String name;
    private String Category;
    private String ProductType;
}
```

RESTORY...

Planera Frontend

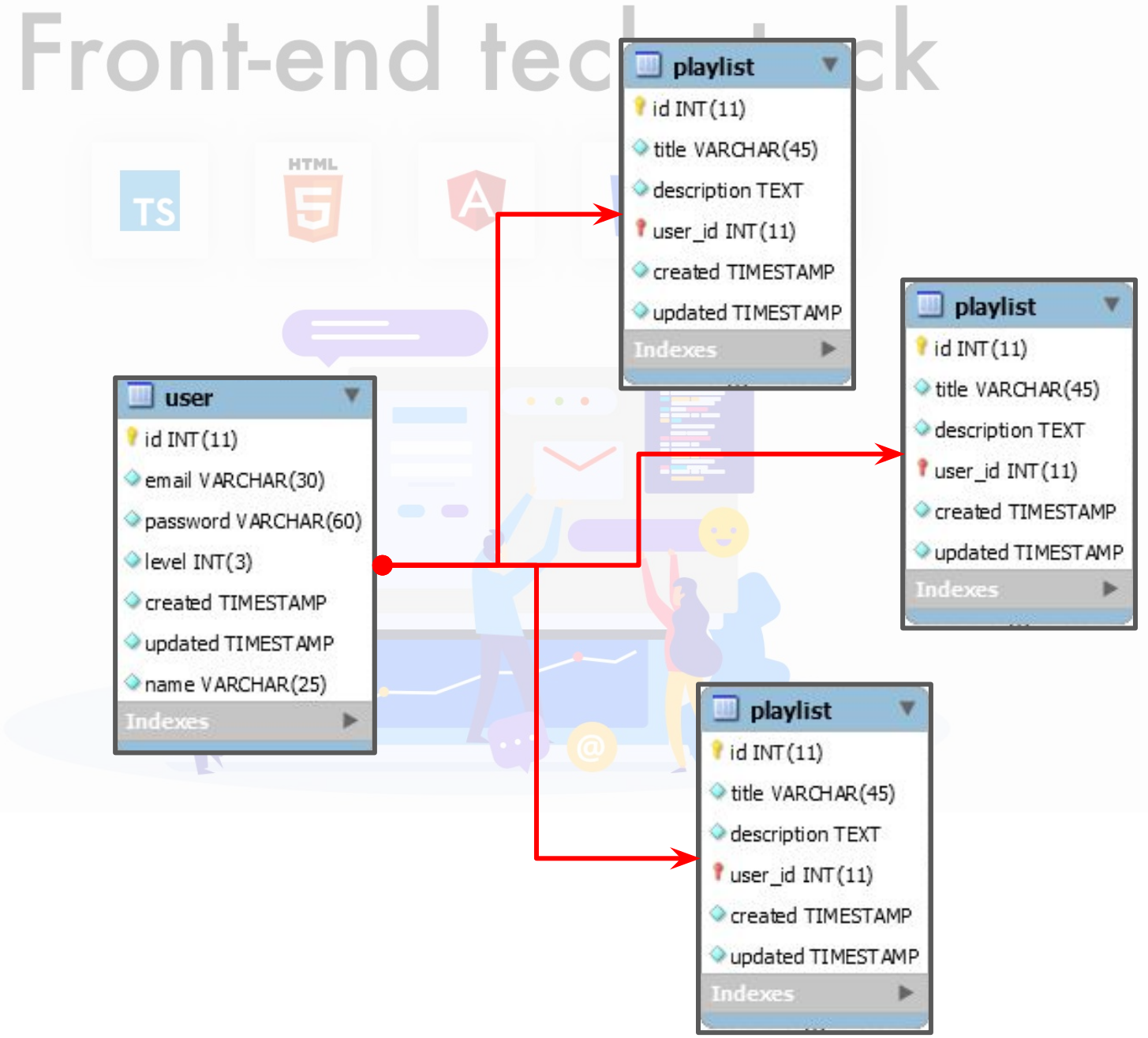
När man ska tänka på hur det ska gå till när de gäller frontend-delen av applikationen det första som ska vara tydligt är designmönstret man ska använda.

Med ett designmönster är det enklare att veta vilka komponenter behövs i applikationen, vilka är statiska och vilka är dynamiska. Hur ska de uppdateras, när ska de uppdateras, vilket beteende de ska ha osv.

Därför handlar frontend planering av en applikation om komponenter. Vi vet att vi kan hantera *State* och *Behaviour*.

Om vi skulle till exempel ta **Movie Recommendations App**en och vi tänker på en List collection då vet vi att det kan vara smart att ha en komponent som hanterar Listornas *State* & *Behaviour*.

När vi vet hur en specifik komponent ska fungera kan vi då bestämma på vilket sätt vi ska göra dem dynamiska och kanske via *Bindings* uppdatera dem när till exempel en **Collection** uppdateras.



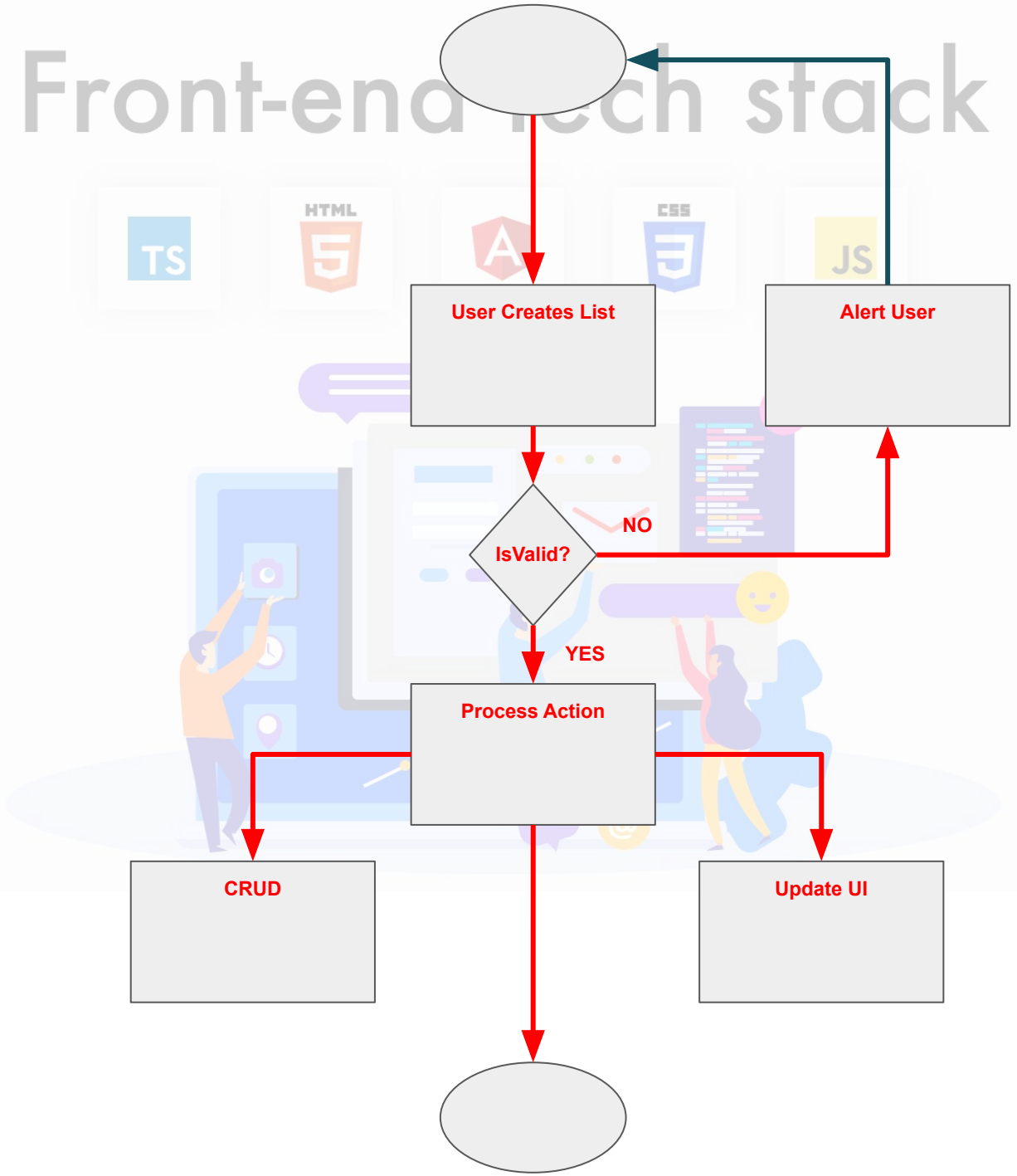
Planera Frontend

FlowCharts är ett bra sätt att visualisera händelseförloppet i ett specifikt system när man försöker implementera en *User Story*.
Ex. Användaren kan skapa en lista med filmer.

För att kunna implementera denna *User Story* korrekt måste vi tänka på de konsekvenser som en sådan *Action* har.

Behöver vi lägga till någon form av validering?
Implementerar vi det på klientsidan eller serversidan?
Hur planerar vi att hantera fel användar*input*?
vad gör vi härnäst?

Ska vi uppdatera användargränssnittet och databasen?



Record Book Application

Record Book Application (Java)

RESTORY...

Inledning

I den här uppgiften ska man ta fram en projektplan som ska levereras med följande:

- UML class -diagram
- Beteendebeskrivningar
- Fungerande Applikation (mer info i nästa slide)

Syftet med projektet är att tillämpa kunskaper från objektorienterad programmering.

RESTORY...

Fungerande Applikation:

Arkitekturen att användas vid detta projekt är valfri, tänk på följande:
Följande “*Model*” klasser ska existera i applikationen.

- RecordCatalogue, RecordBook, Student (*kanske något mer*).

Diskutera i grupp vilka member variables som ska behövas för varje *Model* klass

RESTORY...

Vad handlar uppgiften om...

I den här applikationen ska en lärare kunna hantera data som denne själv skriver i en textfil. När applikationen körs kommer de finnas metoder som läser filen och konverterar “texten” i filen till objekt av följande typer: *Student*, *RecordBook*.

Specs:

- *Läraren ska kunna se info i konsolen om alla studenter som existerar i textfilen och som konverterats till java objekt.*
- *Läraren ska kunna se deras betyg när denne väljer ett specifikt namn.*
- *Metoder som hanterar statistik ska också läraren kunna använda för att se följande:* ***AverageGrade****,* ***HighestGrade****,* ***LowestGrade****.*
- *När applikationen körs och textfilen har hanterats, kommer läraren kunna välja vilken metod han vill använda.*

Dokumentation.

1. Skapa en grundläggande struktur för dokumentet som ska lämnas in individuellt.
2. **Inledning** ska ha en beskrivning av arbetet som ska göras:
   a. Inkluderar en beskrivning av projektet och målet.
   b. User Stories
   c. Teknologierna som ska användas. (Java)
3. En MVP ska identifieras och dokumenteras, inkluderar:
   a. Model klasser som ska användas
   b. UML Class diagrams
   c. Beteendebeskrivningar
4. **Milestones** och **Tasks** ska dokumenteras, inkluderar:
   a. Beskrivning av *milestones*.
   b. Tasks för varje *milestone*.

RESTORY...

Dokumentation.

1. Dokumentera problem och länka dessa problem med specifika *Tasks*.
2. Skapa en Github Repo med *README* som beskriver hur man ska använda produkten.

RESTORY...

User Stories

Baserad på specifikationerna ovan producera en lista med *User stories* som vi diskuterar idag på eftermiddag.
En grundläggande *layout* av planerings dokumentet ska också lämnas in senast idag.
Länk till **Google Docs** funkar

RESTORY...

Redovisning (12/11) & Inlämning (14/11)

1. **Redovisning**:
   - Ni ska diskutera arbetet ni gjort genom att fokusera på målet, resultat och *milestones*
2. **Inlämning**: Producera ett dokument med följande information:
   - Länk till GitHub-repository
   - Dokumentation

Bedömning

1. Denna uppgift kan ge betyg G
   - För G måste samtliga krav i “specifikationerna” slide ha uppnåtts.

Grupp 1
Idris Ahmed
Zaid Al Khateeb
Torun Flink
Troy Fridhult
Ismail Güven

Grupp 2
Muhammed Kan
Fredrik Karmsten
Safer Kret
Andreas Lanhede
Torsten Lareke
Daniel Lindberg

Grupp 3
Josip Marijic
Marcus Montin
Noah Nemhed
Robin Nilsson
Linda Pettersson
Christian Tallner

Grupp 4
William Pihl
Ali Rahbari
Nenad Sekulic
Mohammad Shahriyari
Hassan Siala
Viktor Vallmark

RESTORY...

KÄLLOR

https://3e8.io/2016/questions-when-planning-backend/

https://www.freecodecamp.org/news/have-an-idea-want-to-build-a-product-from-scratch-heres-a-checklist-of-things-you-should-go-through-in-your-backend-software-architecture/

https://medium.com/@ericwindmill/step-by-step-planning-a-web-application-ddaa010a8353

https://www.ishir.com/blog/4585/agile-development-ensure-successful-requirements-gathering-phase.htm

https://dev.to/thecodepixi/fullstack-project-planning-3jml

https://www.freecodecamp.org/news/full-stack-unified-architecture/

RESTORY...