



AMONG ART

PROGETTO FINALE CORSO DI TECNOLOGIE
SEMANTICHE

MEMBRI DEL GRUPPO

RAFFAELE MOLINARI 0622701093 - COORDINATORE

SALVATORE RADIO 0622701217 - STUDIO DEL PROBLEMA

ALESSANDRO ADINOLFI 0622701075 - ARCHITETTURA DEL SISTEMA

GERARDO SALVATI 0622701113 - IMPLEMENTAZIONE

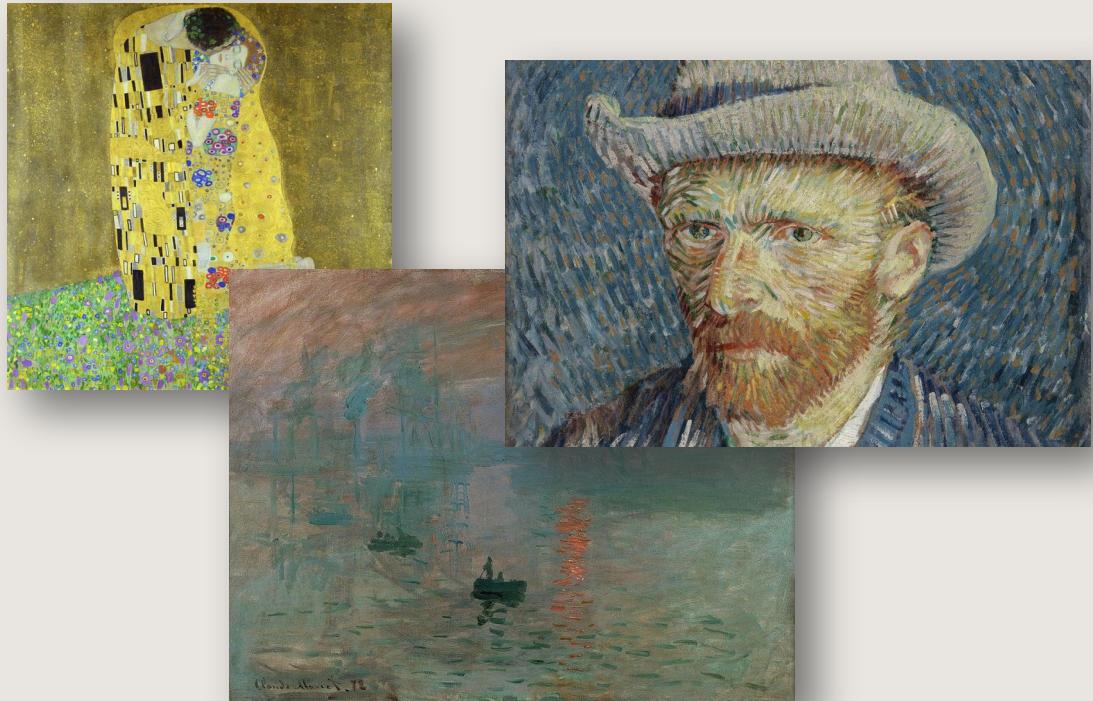


STUDIO DEL PROBLEMA

Si vuole realizzare una web app per ricercare e visualizzare in forma grafica informazioni su quadri ricercati dall' utente.

Informazioni di interesse

- ❖ Nome del dipinto
- ❖ Immagine del dipinto
- ❖ Data di creazione del dipinto
- ❖ Descrizione del dipinto
- ❖ Autore del dipinto
- ❖ Descrizione dell' autore



STUDIO DEL PROBLEMA - ANALISI DEI DATI

Le informazioni di interesse vengono prelevate da due tra le knowledge base più conosciute nel mondo del Semantic Web:

1. Wikidata: è la knowledge base che abbiamo interrogato per ottenere i dati più importanti riguardanti le opere o gli autori.
2. DbPedia: è la knowledge base che abbiamo interrogato per ottenere le descrizioni riguardanti le opere selezionate e i relativi autori.

Poiché i dati nelle knowledge base sono presentati come delle triple, per interrogarle abbiamo utilizzato il linguaggio SPARQL.

SPARQL consente la costruzione di query basate su triple patterns, congiunzioni logiche, disgiunzioni logiche, e pattern opzionali.



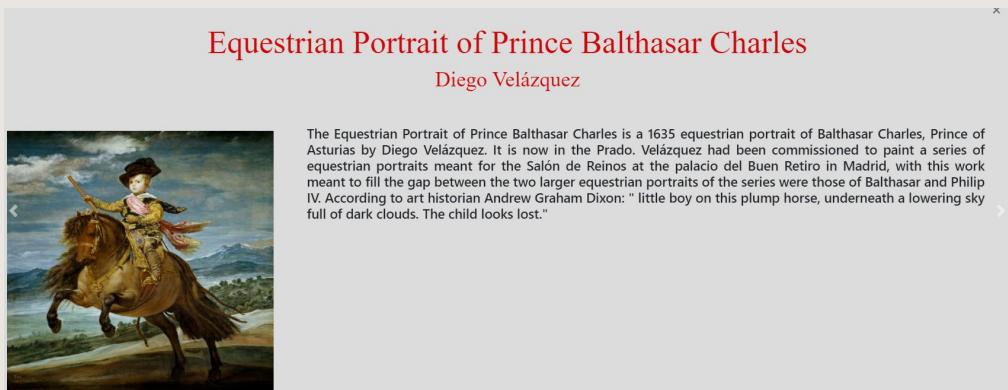
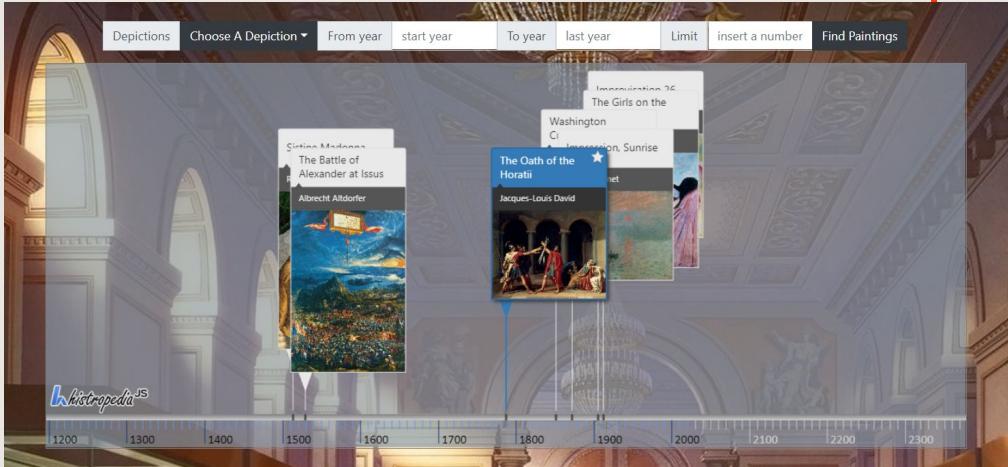
FUNZIONALITA'

- 01 SELEZIONE DI UN' OPERA PER NOME
- 02 TIMELINE
- 03 VISUALIZZAZIONE OPERE PIU' FAMOSE
- 04 VISUALIZZAZIONE CAPOLAVORI ARTISTA
- 05 RICERCA OPERE PER SOGGETTO
- 06 DESCRIZIONE OPERA/AUTORE

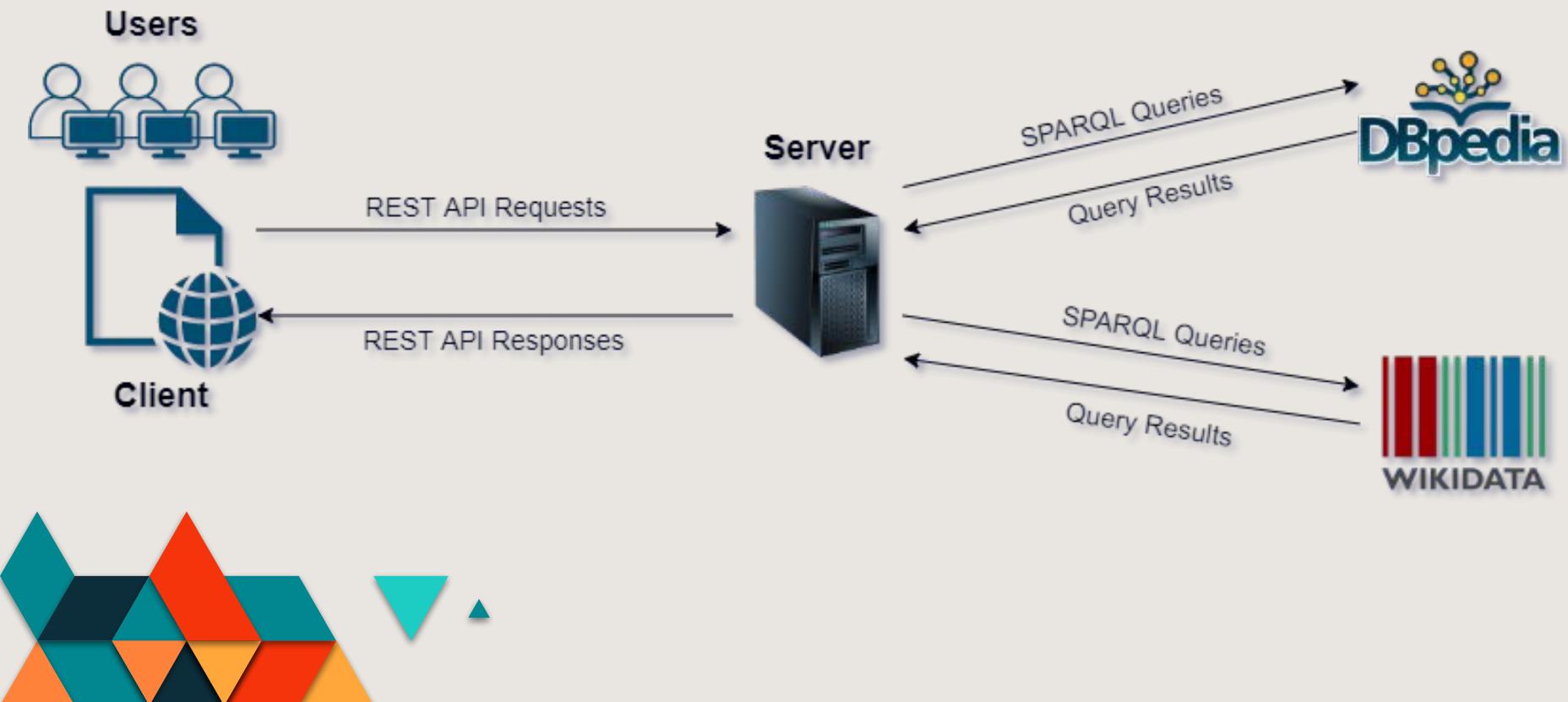
STUDIO DEL PROBLEMA - FUNZIONALITÀ

- ❖ Per ottenere un'opera l'utente può inserire il suo nome completo in un campo di testo. Col riempimento del campo di testo vengono mostrati dei suggerimenti del tipo *NomeQuadro*, *NomeAutore* sfruttando una funzione di autocompletamento.
- ❖ Tutte le opere vengono visualizzate su una timeline tramite delle card contenenti: *NomeQuadro*, *NomeAutore*, *DataDiCreazione*, *ImmagineQuadro*.
- ❖ Vengono mostrate 6 tra le opere più famose al mondo, che fungono da spunto per l'utente.
- ❖ L'utente può visualizzare i capolavori dell'autore di un quadro selezionato e confrontarli.
- ❖ L'utente può aggiungere alla timeline opere contenenti un certo soggetto, scelto tra quelli contenuti nel quadro selezionato, comprese in un certo periodo temporale.
- ❖ L'utente può ottenere la descrizione di un'opera e del suo autore con un doppio click sulla card ad essa associata sulla timeline.





ARCHITETTURA DI SISTEMA



TECNOLOGIE UTILIZZATE - CLIENT E CONDIVISIONE CODICE

Lato Client:

- ❖ HTML5 per definire la homepage
- ❖ Javascript per la creazione di effetti dinamici interattivi
- ❖ CSS per la componente grafica
- ❖ Bootstrap
- ❖ JQuery / JQuery-UI

Condivisione del codice:

- ❖ Fork come client git
- ❖ GitHub

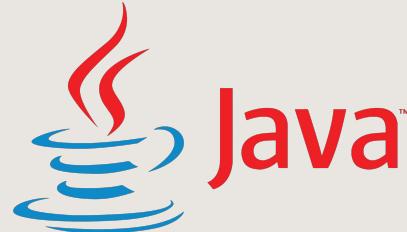
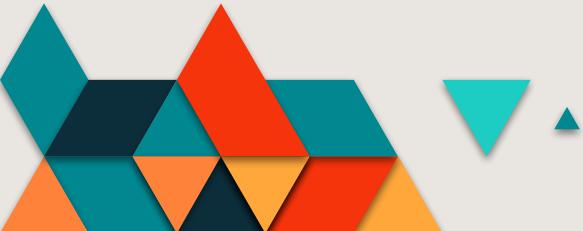


GitHub



TECNOLOGIE UTILIZZATE - SERVER

- ❖ Framework SpringBoot per eseguire applicazioni scritte in Java
- ❖ Postman per simulare le chiamate API da Wikidata e DbPedia
- ❖ Swagger per testare in localHost le API implementate
- ❖ jsonschema2Pojo per definire classi Java da associare alla risposta Json proveniente dalle chiamate API di Wikidata e Dbpedia



COMUNICAZIONE CLIENT - SERVER

Il Client per comunicare col server sfrutta le seguenti API:

- ❖ **GET /user/get-name-painting:** per l'autocompletamento della barra di ricerca.
INPUT: NomeQuadro -> OUTPUT: oggetto JSON
- ❖ **GET /user/get-data-work:** per aggiungere un'opera alla timeline e per creare la descrizione dell'opera e del suo autore.
INPUT: IDWikidataQuadro -> OUTPUT: oggetto JSON
- ❖ **GET /user/get-famous-works:** per suggerire all'utente 6 tra i quadri più famosi.
INPUT: None -> OUTPUT: oggetto JSON



COMUNICAZIONE CLIENT - SERVER

- ❖ **GET /user/get-notable-work:** per aggiungere i capolavori di un autore sulla timeline.
INPUT: IDWikidataQuadro -> **OUTPUT:** oggetto JSON
- ❖ **GET /user/depicts-work:** per ottenere i soggetti contenuti in un quadro
INPUT: IDWikidataQuadro -> **OUTPUT:** oggetto JSON
- ❖ **POST /user/similar-work:** per confrontare un certo numero di quadri contenenti uno stesso soggetto, appartenenti ad un dato periodo temporale.
INPUT: IDWikidataDepict, AnnoDiInizioPeriodo, AnnoDiFinePeriodo, MaxNumeroDiOpereDaAggiungere -> **OUTPUT:** oggetto JSON



FILE STRUCTURE - CLIENT

I file sono organizzati secondo la struttura mostrata nell'immagine.

I file principali sono i seguenti:

- ❖ **AmongArt.html**: è la pagina web con cui si interfaccia l'utente.
- ❖ **AmongArt.css**: è il file CSS con cui abbiamo definito l'aspetto grafico della nostra pagina.
- ❖ **AmongArt.js**: è lo script da noi definito per personalizzare l'interazione dell'utente con la pagina.



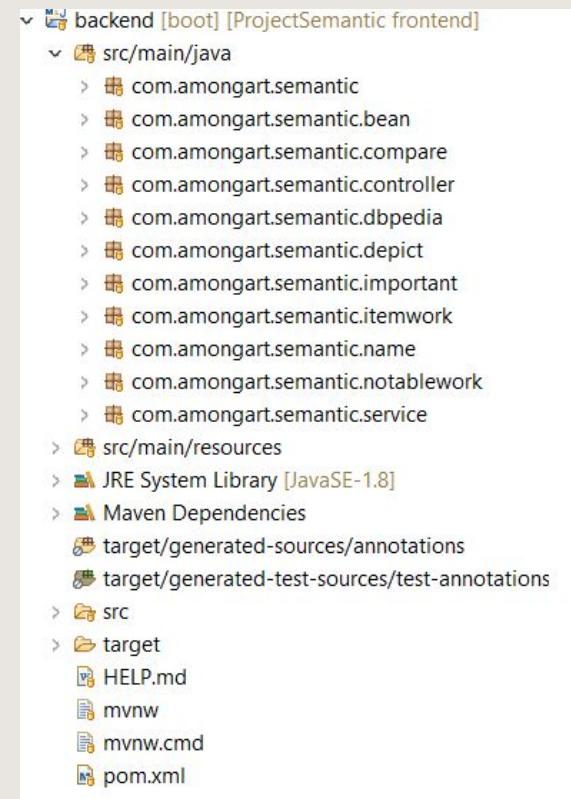
FILE STRUCTURE - SERVER

Package controller: contiene la classe che mette a disposizione le API per il frontend

Package service: viene definita la classe e la sua relativa interfaccia, che permette di chiamare e manipolare i dati di Wikidata e DbPedia

E' stato creato un package per ciascuna API messa a disposizione.

All' interno di ciascun package troviamo gli oggetti java che mappano la risposta di Wikidata o DbPedia, più la risposta di output per il frontend, generata nel package service



IMPLEMENTAZIONE - CHIAMATE AJAX LATO CLIENT

```
onArticleDoubleClick: function(article) {  
  
    $("#description-container").remove();  
    $("#timeline-container").after('<div id="description-container" class="carousel slide" data-ride="carousel" style="margin-top: 20px;">  
        .ajax({  
            type: "GET",  
            url: "http://localhost:8080/user/get-data-work?idNameWorkWikidate=" + article.data.idItem,  
            success: function(resp){  
  
                //Description painting container  
                $("#description-container").append('<button style="padding: 10px 24px;" type="button" id="close-description" class="close" data-dismiss="modal"></button>');  
                $("#description-container").append('<div id="carousel-inner" class="carousel-inner">');  
                $("#carousel-inner").append('<div id="carousel-item-active-1" class="carousel-item active">');  
                $("#carousel-item-active-1").append('<h1 id = "title-painting"></h1>');  
                $("#carousel-item-active-1").append('<h2 id = "name-creator"></h2>');  
                $("#carousel-item-active-1").append('<img id = "image-painting" style="float:left;margin:50px 50px 20px 15px" width="100px" height="100px"/>');  
                $("#carousel-item-active-1").append('<h5 style="margin:50px 50px 50px 50px" id = "description-painting"></h5>');  
                $('#title-painting').text(resp.nameItem);  
                $('#name-creator').text(resp.nameCreator);  
                $('#image-painting').attr("src", resp.imageItem);  
  
                var description = "Description not found"  
  
                if (resp.descriptionItem == null) {  
                    $('#description-painting').text(description);  
                } else {  
                    $('#description-painting').text(resp.descriptionItem);  
                }  
            }  
        })  
    });  
};
```



IMPLEMENTAZIONE - EFFETTUARE UNA QUERY DAL SERVER

```
@Override
public ResponseWork getWork(String idWikidataWork) throws ClientProtocolException, IOException {
    HttpClient httpClient = HttpClientBuilder.create().build();
    //%2C virgola
    //%20 spazio
    idWikidataWork = idWikidataWork.replace(",","%2C");
    idWikidataWork = idWikidataWork.replace(" ","%20");
    HttpGet httpRequest = new HttpGet("https://query.wikidata.org/sparql?query=SELECT%20DISTINCT%20%3Flabel%20%3Fauth");
    Gson gson = new Gson();

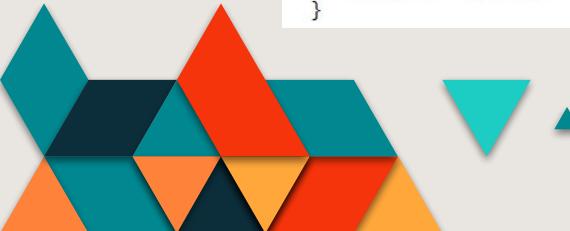
    httpRequest.addHeader("Accept", String.valueOf(ContentType.APPLICATION_JSON));
    HttpResponse httpResponse = httpClient.execute(httpRequest);

    HttpEntity httpResponseEntity = httpResponse.getEntity();
    ResultWork wikidataWorkResult = gson.fromJson(EntityUtils.toString(httpResponseEntity),
        ResultWork.class);

    ResponseWork response = getResponse(wikidataWorkResult);
    response.setIdWikidataItem(idWikidataWork);
    ResponseDbpedia responseDbpedia = getDescription(response.getIdWikidataCreator(), response.getIdWikidataItem());

    response.setDescriptionCreator(getDescriptionAuthor(responseDbpedia));
    response.setDescriptionItem(getDescriptionItem(responseDbpedia));

    return response;
}
```



VERSIONE FINALE - PROBLEMI APERTI E CHIUSI

Problemi risolti:

- ❖ Convertire il risultato della query SPARQL in un oggetto JAVA
- ❖ Rimozione duplicati dei risultati di una query aventi la stessa data
- ❖ Tempo di attesa alto per il risultato delle query
- ❖ Comunicazione tra lato client e server dell'applicazione
- ❖ Eseguire chiamate SPARQL su Wikidata lato server
- ❖ Rimozione quadri famosi non aventi foto

Problemi aperti:

- ❖ Descrizione mancante dei quadri che non hanno una risorsa associata su DbPedia
- ❖ Rimozione duplicati aventi la stessa immagine e lo stesso autore



**GRAZIE PER L'
ATTENZIONE**

