Progetto1

Giulio Bosco - Progetto 1

Introduzione

Informazioni sul progetto

Allievo: Giulio Bosco
Docenti: Luca Muggiasca, Adriano Barchi, Francesco Mussi, Elisa Nannini
Scuola: Scuola Arti e Mestieri di Trevano
Sezione: SAMT I3AA
Data inizio: 05.09.2018
Data fine: 09.11.2018 Git-Hub link: https://github.com/boscogiulio/progetto1

Abstract

> Is required an web application for collect the basic information of people. Used java for write the collected data on the CSV file in the server, as web-server has been used Apache Tomcat, written in Java, so all the back-end stuffs are written in Java.

Scopo

Questa è una webapp a scopo didattico. L'applicazione deve permettere gli utenti di registrarsi con i propri dati. Deve salvare le registrazioni in due file CSV. Uno deve contenere i dati giornalieri mentre il secondo tutte le registrazioni. I dati devono essere validati.

Analisi

Analisi del dominio

L'applicazione servirà per raccogliere dati di persone sconosciute si presume che gli utenti non abbiano alcuna competenza informatica, l'applicazione potrebbe venire utilizzata sa qualunque dispositivo che possa avere accesso ad internet, quindi:

- Dispositivi Mobile (Smartphone / Tablet)
- Dispositivi Desktop

Analisi e specifica dei requisiti

ID	**REQ-001**
Nome	Dati
Priorità	1
Versione	1.0
Note	File CSV
	Sotto requisiti
001	Salvare i file in 2 CSV (Registrazioni_tutte.csv, Registrazione_yyyy_mm_dd.csv)
002	File salvati nella root del sito sotto "Registrazioni"
003	separatore ";"
004	permessi di scrittura

ID	**REQ-002**
Nome	Pagine
Priorità	2
Versione	1.0
Note	ci devono essere 4 pagine.
	Sotto requisiti
001	ci deve essere una pagina con una descrizione del prodotto: introduzione
002	ci deve essere una pagina di inserimento dei dati.
003	ci deve essere una pagina per controllare i dati precedentemente inseriti.
004	ci deve essere una pagina per visualizzare i dati appena inseriti.

ID	**REQ-003**
Nome	Pagina *Introduzione*
Priorità	2
Versione	1.0
Note	pagina di benventuto
	Sotto requisiti
001	Ci deve essere un tasto che porta alla pagina di inserimento dei dati.

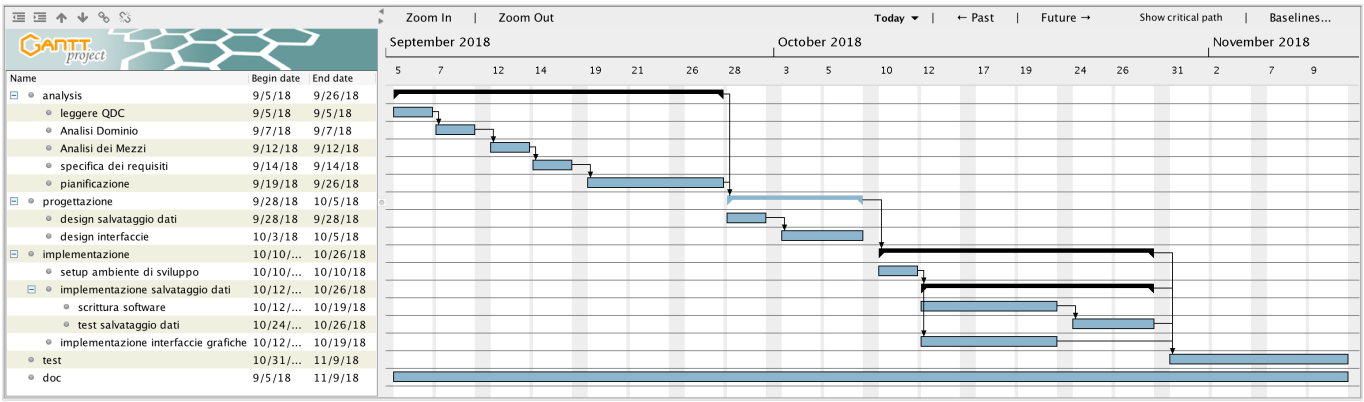
ID	**REQ-004**
Nome	Pagina *Inserzione Dati*
Priorità	1
Versione	1.0
Note	Form dei dati
	Sotto requisiti
001	Validazione dei dati
002	Ci deve essere un tasto che annulli l'inserimento fatto, cancelli tutti i contenuti dei campi.
003	Ci deve essere un tasto che porti alla pagina di controllo dei dati.

ID	**REQ-005**
Nome	Pagina *Controllo Dati*
Priorità	1
Versione	1.0
Note	Controllare i dati
	Sotto requisiti
001	Mostrare dati come nel form di registrazione
002	Ci deve essere un tasto per correggere i dati inseriti, che riporti alla pagina: *Inserzione Dati*
003	Ci deve essere un tasto per la registrazione, che scriva i dati sul csv e che poi porti alla pagina di *Lettura dei dati dal CSV*

ID	REQ-006
Nome	Pagina _ Lettura dati da CSV_
Priorità	2
Versione	1.0
Note	presentazione dati
	Sotto requisiti
001	la grafica di questa pagina deve essere uguale alla pagina di registrazione
002	la posizione dei campi deve essere uguale alla pagina di registrazione.
002	dati letti da "Registrazione_yyyy_mm_dd.csv"
003	Teasto per ritornare alla pagina benvenuto

ID	REQ-007
Nome	Grafica
Priorità	3
Versione	1.0
Note	La grafica delle pagine è a discrezione del esecutore.

Pianificazione



Analisi dei mezzi

Software

- Librerie:
  - JavaScript
    - AngularJS - v1.3.5
    - jQuery - v3.2.1
  - CSS
    - Bootstrap - v3.3.4
    - FontAwesome - 4.3.0
  - Java
    - Java Tomcat - v9.0.10
- IDE:
  - jetBrains IntelliJ IDEA
- Progettazione:
  - StarUML
  - Adobe PhotoShop
  - Gantt Project

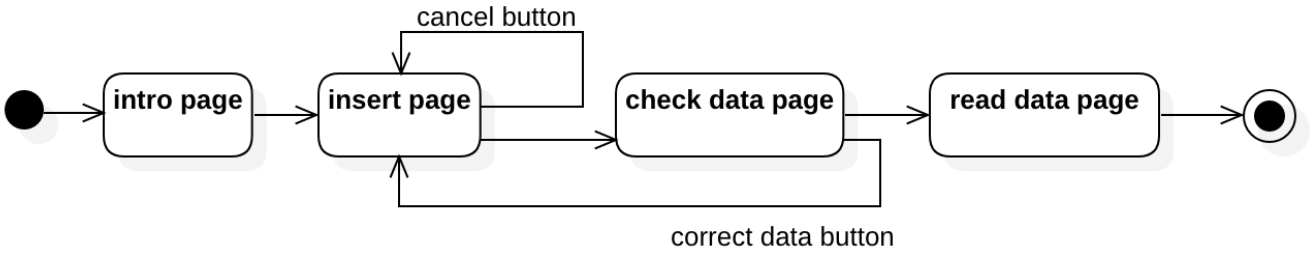
Hardware

- Sviluppo
  - portatile: Apple - MacBookPro 2018
    - Processore: Intel Core i7
    - RAM: 16GB
    - Disk: 1TB SSD

Progettazione

Design dell’architettura del sistema

Quando si entra nella prima pagina vi è un introduzione, dopo vi è una pagina per l'inserimento dei dati, che ha 2 bottoni uno per procedere, il secondo



Design dei dati e database

Salvataggio dei dati eseguito su due file CSV.

- Total CSV data file:
  - Contiene tutti i record
  - Path: "/Registrazioni/Registrazioni_tutte.csv"
- Daily CSV data file:
  - Contiene le registrazioni di un giorno.
  - Ogni giorno viene creato un nuovo file.
  - Path: "/Registrazioni/Registrazione_YYYY_MM_DD.csv"

Struttura dei dati:

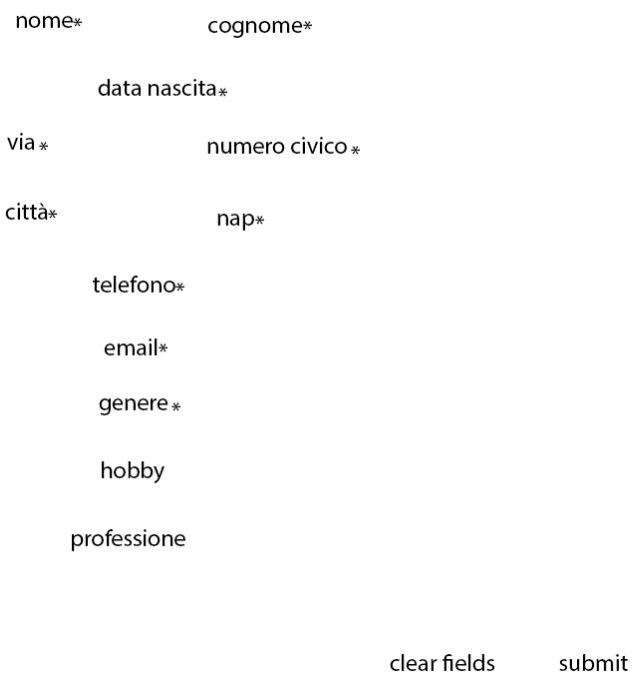
nome campo	**tipo campo**
data-ora	s
nome	testo
cognome	test
data_di_nascita	data
via	testo
numero civico	numero 3 cifre
città	testo
nap	numero 5 cifre
numero di telefono	testo (sole cifre, spazi, trattini)
e-mail	testo (controllo formato email)
genere	testo (solo M o F)
hobby	testo
professione	testo

Design delle interfacce

Pagina di introduzione

In questa pagina vi è una introduzione al sito.
Con un bottone per spostarsi sulla pagina di inserimento dei dati.

Pagina inserimento dati

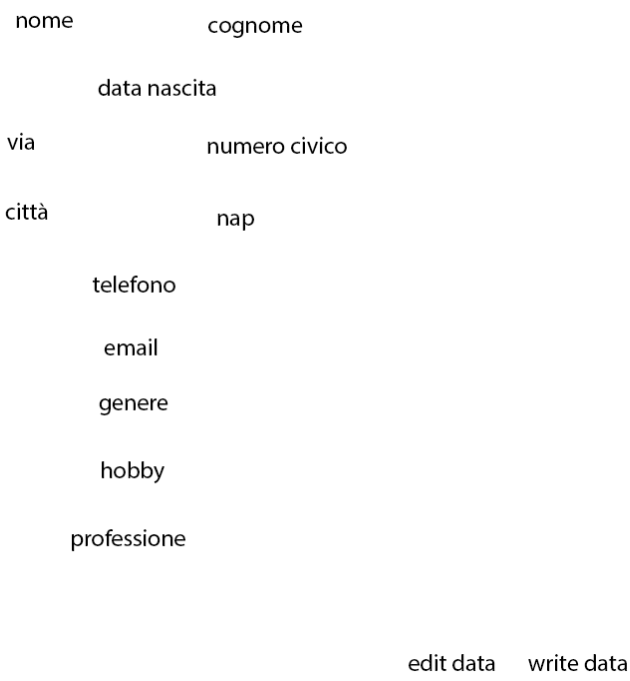


In questa pagina vi sono i campi per inserire i dati.
Ci sono i seguenti campi di input, (quelli con * sono obbligatori):

- Nome*
- Cognome*
- Data di nascita*
- Indirizzo
    - Cia*
    - Numero civico*
    - Cap*
    - Città*
- Numero di telefono*
- Indirizzo e-mail*
- Hobby
- Professione

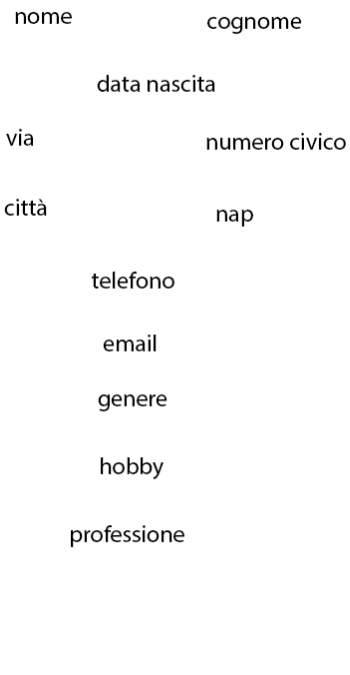
Ci saranno anche due bottoni, uno per procedere ed uno per resettare tutti i campi.

Pagina controllo dei dati



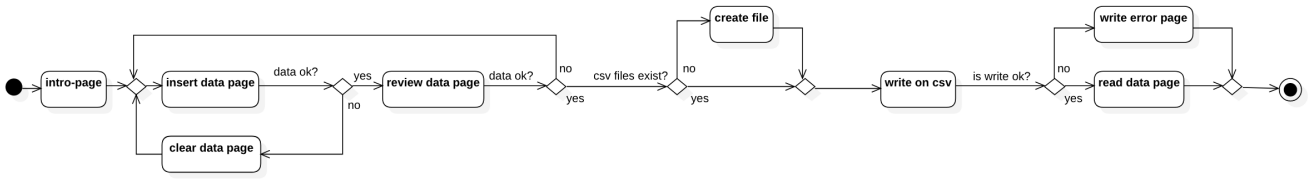
Nella pagina di controllo dei dati, vi sarnno gli stessi campi che nella precedente.
Vi saranno sempre due bottoni, ma con due funzionalità diverse, uno scriverà i dati nel file CSV mentre il secondo permetterà di tornare alla pagina precedente per modificare i dati inseriti.

Pagina di lettura dei dati



Nella pagina di lettura dei dati, vi sarà la possibilità di leggere l'ultimo dato inserito, o i dati inseriti nella giornata corrente (per orario del server), oppure tutti i dati inseriti.
Questo sarà possibile tramite te sezioni, con un menu sopra le quali.

Design procedurale



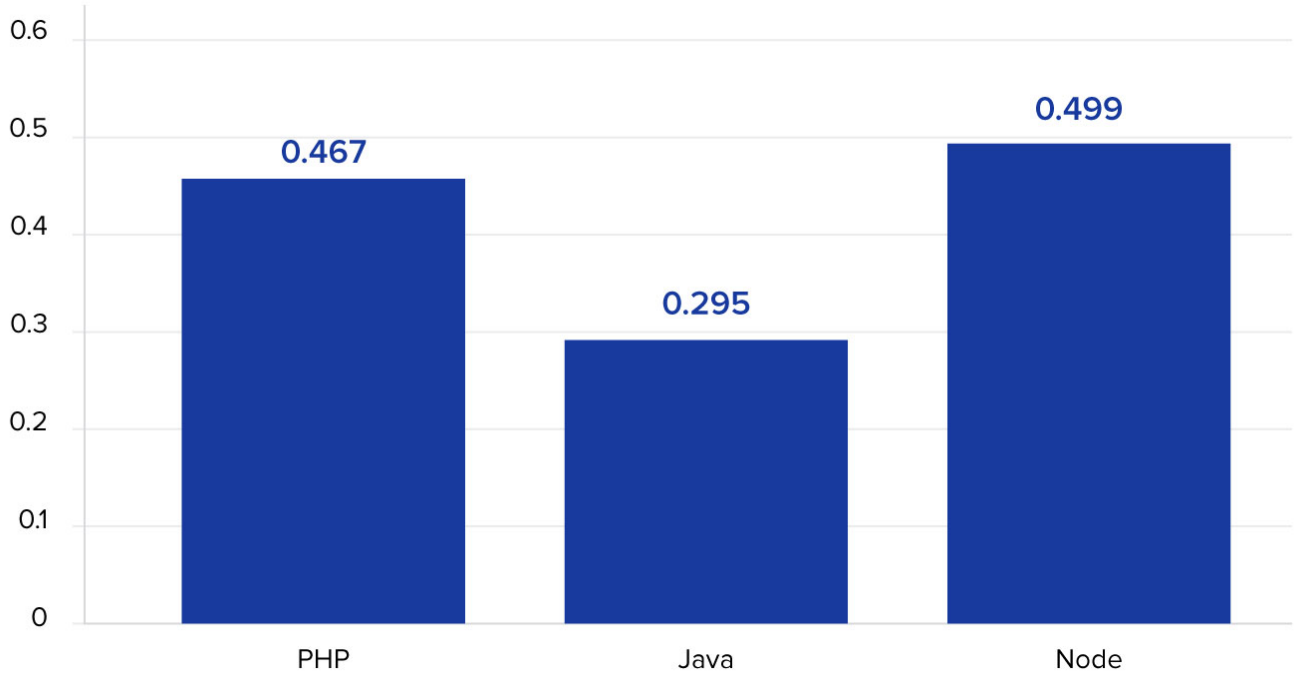
Entrando nel sito ci sarà una pagina di introduzione, poi la pagina di inserimento dati, dalla quale si puo procedere oppure resettare tutti i campi. Proseguendo vi è la pagina di controllo dei dati, dalla quale si può procedere oppure ritornare sull'inserimento dei dati e modificare i nomi. Procedendo si va sulla pagina di lettura dei dati.

Implementazione

Web Server

Per implementare la parte back-end di questo progetto (cioè la parte lato server), vi erano alcune opsioni di linguaggi di programmazione o scripting per eseguire le operazioni di lettura e scrittura sui file:

- PHP
- Java
- NodeJS



toptal.com

In questo grafico è mostrato il tempo di risposta per una richiesta a parità di prestazioni del server.
Si può notare che Java è nettamente più veloce di PHP (questo accade soprattutto quando bisogna eseguire degli algoritmi o operazioni complicate). Questa grande differenza sta nel fatto che Java viene compilato e non interpretato come PHP. È stato deciso di non utilizzare NodeJS perché non si hanno nemmeno le conoscenze di base sul suo funzionamento.

Per questo progetto la differenza è minima e non verrebbe neanche notato dall'utente finale se viene usato un sistema o l'altro. Ho deciso di utilizzare comunque Java (non ostante ero a conoscenza che avrei riscontrato più difficoltà nello sviluppo dell'applicativo; Ed avrei speso molto tempo a capire come funziona il web server in Java). Ho preso questa decisione per iniziare a prendere confidenza con questo sistema, che potrebbe essermi molto utile in progetti più grandi ed importanti dove si ricerca il massimo della prestazione.

Configurazione del web-server

Scaricare ed installare la Java Virtual Machine (JRE - Java Runtime Environment) e Java Development Kit (JDK).

```
http://www.oracle.com/technetwork/java/javase/downloads/index.html
```

Scariare i file binari di Tomcat:

```
http://tomcat.apache.org/
```

Per configurare le variabili d'ambiente di tomcat si può usare lo script fornito assieme hai file binari oppure manualmente (Guida per installazione manuale fornita assieme a tomcat). Lo script è disponibile sia per UNIX (`setevn.sh`), che per windows (`setevn.bat`), in entrambi i casi va abilitato.

Ambiente di sviluppo

JetBrains IntelliJ IDEA, un tool professionale per sviluppo in Java. Utilizzandolo con la licenza "education" è gratuito. Siccome l'applicativo non verrà venduto, si ha il diritto di utilizzare questa licenza.

Installazione IntelliJ IDEA

Scaricare il file di installazione dal sito ufficiale ed avviarlo, poi seguire la procedura guidata.

Configurazione Apache Tomcat con IntelliJ IDEA

Prima di tutto bisogna configurare il web server Apache Tomcat (Capitolo precedente).

Poi creare il progetto seguendo la procedura guidata per creare un progetto di tipo: `Java Enterprise > Web Application` dopo di che aggiungere una configurazione di web server, scegliere il tipo `Tomcat > Local`, poi cliccare su `Configure`, li aggiungere un server inserendo il suo nome ed il percorso dei file binari precedentemente scaricati.

Sviluppo applicativo

Validazione dei dati

La validazione dei dati è fatta sia in front-end che in back-end, questo per garantire velocità e prestazioni (front-end) e sicurezza dell'inserimento corretto dei dati (back-end).

Struttura web

```
 --- /index.html            (Pagina introduzione)
  |
  |- /insert.html           (Pagina inserimento dati)
  |
  |- /Insert                (Java InsertsServlet - Controllo inserimento
dei dati)
  |
  |- /check.jsp             (Pagina di controllo dei dati)
  |
  |- /edit.jsp              (Pagina di modifica dei dati)
  |
  |- /Save                  (Java SaveServlet - Salvataggio dei dati)
  |
  |- /read.html             (Pagina di lettura dei dati)
   |- assets/data/last.jsp  (Ultimo record in formato JSON)
   |- assets/data/today.jsp (Registrazioni con la data odierna del server
in formato JSON)
   |- assets/data/all.jsp   (Tutte le registrazioni in formato JSON)
```

Front-End

Per sviluppare le interfacce grafiche è stato utilizzato HTML & CSS (e Bootstrap), per la validazione dei dati JavaScript con la libreria jQuery e per interpretare i dati AngularJS (nella pagina di lettura dei dati).

Introduzione

La pagina di introduzione è il file: `index.html`.

Contiene il titolo della pagina ed un link alla pagina di inserimento dati.

Inserimento dati

La pagina di inserimento dei dati è il file: `insert.html`.

La pagina di inserimento dati rispetta l'ordine dei campi come nel design delle interfaccie grafiche, sono stati migliorati gli aspetti grafici rispetto a quello che si pensava inizialmente.

Quando un campo non è completato correttamente si colora il testo di rosso ed anche il bordo inferiore del elemento di input.

Inizialmente per il campo della data si voleva inserire un campo di inserimento data con il calendario, poi pensando che dovendo inserire delle date di nascita, bisogna scorrere tutti i mesi di tutti gli anni, questo processo rischia di diventare molto lungo. Quindi è meglio un campo per il giorno, uno per il mese ed uno per l'anno.

Quando tutti i campi obbligatori sono riempiti correttamente il bottone `procedi` si sblocca. È presente anche un'altro bottone che resetta tutti i valori.

Invia i dati con il metodo POST alla JavaServlet `SaveServlet` identificata lato web con la definizione di accesso `/Save`.

Validazione dei dati
La validazione dei dati è eseguita con dei validator scritti in JavaScript ed utilizzando jQuery per selezionare i campi da controllare, gestire i valori e gestire i colori (del testo e del bordo).
Le validazioni sono fatte con le espressioni regolari, come segue (esempio di espressione regolare per validare un numero telefonico):

```
/**
 * Validate the phone number.
 *
 * @param phone {string} Number to validate.
 * @returns {boolean} Result of the validation.
 */
function phone(phone) {
        let regex = /^\d+$/;
        if (phone.trim().length > 8 && phone.trim().length < 13 &&
regex.test(phone)) {
                return true;
        } else {
            return false;
        }
}
```

Espressione regolare: `/^\d+$/`, indica che accetta solo numeri.

Controllo dei dati

La pagina del controllo dei dati è sul file: `check.jsp`.

La pagina è la stessa di quella dell'inserimento dei dati, con la differenza che i campi sono bloccati, quindi i valori non sono modificabili. Anche i due bottoni hanno funzionalità diverse, uno serve per richiamare la pagina di modifica dei dati, mentre il secondo per scrivere sul file.

Modifica dei dati

La pagina di modifica dei dati è sul file: `edit.jsp`.

La pagina è quella di introduzione, solamente che prende i valori dalla sessione e li inserisce già nei campi. Così da essere pronti per essere modificati.

La pagina di lettura dei dati

La pagina di lettura dei dati è sul file `read.html`.

La pagina di lettura dei dati è composta dal titolo della pagina, una barra di navigazione con 3 opsioni, ultimo record, record odierni e tutti i record. Ogniuna di queste voci avrà una sezione (solo quella selezionata sarà visibile).

Nella prima sarà visibile subito tutto il contenuto della registrazione; mentre nelle seconde due sarà visibile subito il nome, il cognome e la data di nascita. cliccandovi sopra si potranno vedere tutti i dati.

I dati vengono richiesti al server tramite una richiesta AJAX, il server ritorna un file JSON (uno per ogni sezione), che verrà interpretato da AngularJS.

Inizializzazione dell'AngularJS WebApp:

```
let app = angular.module('ReadApp',[]);

app.controller('LastCtrl', ['$scope', '$http', function ($scope, $http) {
        $http.get('assets/data/last.jsp')
                .then(function (response) {

                        let data = response.data;

                        $scope.lastSubs = data;
                });
}]);

app.controller('TodayCtrl', ['$scope', '$http', function ($scope, $http) {
        $http.get('assets/data/today.jsp')
                .then(function (response) {
                        let data = response.data;

                        $scope.todaySubs = data;
                })
}]);

app.controller('AllCtrl', ['$scope', '$http', function ($scope, $http) {
        $http.get('assets/data/all.jsp')
                .then(function (response) {
                        let data = response.data;

                        $scope.allSubs = data;
                })
}]);
```

Back-End

Il back-end dell'applicativo è stato sviluppato in Java, utilizzando il web server Apache Tomcat (v9.0.10).

Validators

Come prima cosa sono stati implementati i `validators`, cioè le classi che si occupano della validazione dei dati. I validatori sono:

Nome	Utilizzo
Validator	Validazione di stringa, controllo della lunghezza
DateValidator	Validazione di oggetti data
DomainValidator	Validazione di dominii internet
UsernameValidator	Validazioni di username
EmailValidator	Validazione di indirizzi email (utilizza UsernameValidator e DomainValidator)
IntegerValidator	Validazione di numeri interi
NumberValidator	Controlla il numero di cifre
NameValidator	Validazione di Nomi, controlla che siano tutte lettere (anche con accenti)

CSV & Dati

Dopo i validatori sono state implementate le classi relative alla gestione dei dati, più precisamente le classi relative al CSV e hai record del CSV.

- Csv - Gestione dei file CSV, inserimento e lettura dei dati
  In questa classe engono utilizzati principalmente i paradigmi di programmazione relativi al file system e all'interpretazione dei dati da un CSV.
- CsvToJson - Convertire il formato CSV ad una forma JSON di base
  In questa classe viene utilizzata la classe Csv per interpretare i dati e poi vengono trasformati sotto forma di JSON, in una stringa.
- Address - Indirizzo: Via, Numero civico, Città, CAP, Paese
- Record - Struttura del CSV e dati con relativa validazione
  Struttura dei dati da registrare nell'applicativo
- RecordManager - Gestione dei Record, Lettura o scrittura nel record
  Gestione dei CSV, per raggruppare la logica di gestione dei CSV.

Analisi dei dati

Per facilitare il lavoro di controllo e analisi dei dati sono state create delle classi che analizzano le richieste e le sessioni HTTP.
Controllano che vi siano i dati obbligatori e che siano corretti, poi li prepara per poter essere utilizzati dalla con la classe Record.
Gli analyzer sono strettamente legati al record, quindi per altri utilizzi vanno riviste alcune parti.

Sono stati creati due analyzer, uno per le richieste ed uno per le sessioni:

- RequestAnalyzer
- SessionAnalyzer

JavaServlets & JSP

Le JavaServlets sono classi che vengono richiamate con delle richieste HTTP (o HTTPS) con almeno due metodi:

- `doGet` (viene richiamato quando la richiesta è in get)
- `doPost` (viene richiamato quando la richiesta é in post)

Solitamente non vengono utilizzate per ritornare delle pagine HTML, ma per eseguire operazioni più complesse (Spesso eseguono un redirect su una pagina dinamica o statica). Mentre per ritornare pagine HTML (Pagine dinamiche) vengono utilizzate le JSP (JavaServer Peages).

Le parti più complicate delle JavaServlets e JSPs sono la gestione delle sessioni utilizzata per mantenere i dati. Siccome prima di creare la sessione vengono eseguiti molti controlli è stato deciso di spostare il codice relativo alla sessione negli analyzer.

Esempio di servlet con redirect alla pagina `index.html`.

```
import java.io.IOException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpSession;


@WebServlet(name = "ExampleServlet")
public ExampleServlet extends HttpServlet {
    @Override
    protected void doPost(
            HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        String redirectURL = response.encodeRedirectURL("index.html");
        response.sendRedirect(redirectURL);
    }

    @Override
    protected void doGet(
            HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        this.doPost(request, response);
    }
}
```

Sono state create due Servlet:

- InsertServlet - Servlet per l'inserimento dei dati, esegue i controlli e reindirizza alla pagina di controllo dei dati.
- SaveServlet - Controlla i dati e poi li salva nel file csv.

Sono state create 2 pagine html con JSP.

- check.jsp
- edit.jsp

Sono stati creati 3 file JSON con le JSP.

- assets/data/last.jsp
- assets/data/today.jsp
- assets/data/all.jsp

I file json vengono letti i dati dal file CSV dalla classe `Csv.java` e trasformati in JSON dalla classe `CsvToJson.java` tutto questo processo viene gestito dalla classe `RecordManager.java`

Qui sotto è riportato il codice che genera il file JSON last.jsp:

```
<%@ page import="data.RecordManager" %>
<%@ page import="java.io.IOException" %>
<%@ page import="helper.csv.NoCsvHeaderException" %>
<%
    /*
     * Last subscription in json.
     *
     * @author giuliobosco
     * @version 1.0
     */
%>
<%
    RecordManager rm = new RecordManager();
    String record = "Error";
    try {
        record = rm.getLastRecord();
    } catch (IOException ioe) {

    } catch (NoCsvHeaderException nche) {

    }
%>
<%=record%>
```

Test

Protocollo di test

Test Case	**TC-001**
Nome	Test dati
Riferimento	REQ-001 Sub-REQ-001
Descrizione	Controllare che i file CSV vengano salvati correttamente
Prerequisiti	Il prodotto deve essere completo
Procedura	1. inserire i propri dati nella pagina di registrazione. 2. Cliccare su procedi 3. Controllare di avere inserito i propri dati correttamente e cliccare su salva.
Risultati attesi	Due file nella cartella Registrazioni, uno `Registrazioni_tutte.csv` ed uno `Registrazione_yyyy_mm_dd.csv`

Test Case	**TC-002**
Nome	File salvati nella root del sito sotto `Registrazioni`
Riferimento	REQ-001 Sub-REQ-002, Sub-REQ-003
Descrizione	Controllare che i files CSV vengano salvati nella cartella `Registrazioni`
Prerequisiti	Il prodotto deve essere completo
Procedura	1. inserire i propri dati nella pagina di registrazione. 2. Cliccare su procedi 3. Controllare di avere inserito i propri dati correttamente e cliccare su salva. 4. Aprire in una nuova pagina `/Registrazioni/Registrazioni_tutte.csv` sull'indirizzo del server. 5. Aprire in una nuova pagina `/Registrazioni/Registrazione_yyyy_mm_dd.csv` sull'indirizzo del server.
Risultati attesi	Nelle due pagine aperte vi devono essere i file csv. Uno con tutte le registrazioni, mentre il secondo con le registrazioni giornaliere. All'interno dei csv i dati devono essere separati da un `;`.

Test Case	**TC-003**
Nome	Pagine
Riferimento	REQ-002
Descrizione	Controllare che siano state implementate tutte le pagine
Prerequisiti	Il prodotto deve essere completo
Procedura	1. Controllare che sia sta creata la pagina di benvenuto. 2. Controllare che sia stata creata la pagina di inserimento dati. 3. Controllare che sia stata creata la pagina di controllo dei dati. 4. Controllare che sia stata implementata la pagina di lettura dei dati.
Risultati attesi	Le quattro pagine devono essere presenti.

Test Case	**TC-004**
Nome	Verifica dei campi di input
Riferimento	REQ-004 Sub-REQ-001
Descrizione	Controllare che i campi di input controllino i valori correttamente
Prerequisiti	La pagina di registrazione deve essere implementata
Procedura	1. inserire i propri dati nella pagina di registrazione. 2. verificare che nessun campo venga sottolinato di rosso.
Risultati attesi	Nessun dato deve essere sottolineato di rosso.

Test Case	**TC-005**
Nome	Pagina inserimento dati, Bottone annullamento
Riferimento	REQ-004 Sub-REQ-002
Descrizione	Controllare che il tasto di annullamento annulli tutti i campi.
Prerequisiti	Il prodotto deve essere completo
Procedura	1. Inserire dei dati nei campi. 2. Cliccare su annulla.
Risultati attesi	I campi devono essere vuoi.

Test Case	**TC-006**
Nome	Pagina inserimento dati, Bottone procedi
Riferimento	REQ-004 Sub-REQ-003
Descrizione	Controllare che il tasto di avanzamento porti alla pagina di controllo dei dati.
Prerequisiti	Il prodotto deve essere completo
Procedura	1. Inserire i dati nella pagina di inserimento. 2. Cliccare su procedi.
Risultati attesi	Bisogna essere portati sulla pagina di controllo dei dati con i dati inseriti precedentemente.

Test Case	**TC-007**
Nome	Pagina controllo dati, Bottone modifica
Riferimento	REQ-005 Sub-REQ-002
Descrizione	Controllare che il botton di modifica dei dati funzioni correttamente.
Prerequisiti	Il prodotto deve essere completo
Procedura	1. inserire i propri dati nella pagina di registrazione. 2. Cliccare su procedi 3. Cliccare su modifica.
Risultati attesi	La pagina deve essere quella di inserimento dei dati con i dati già inseriti.

Test Case	**TC-008**
Nome	Pagina controllo dati, Bottone salva
Riferimento	REQ-005 Sub-REQ-003
Descrizione	Controllare che il botton di salvataggio dei dati funzioni correttamente.
Prerequisiti	Il prodotto deve essere completo
Procedura	1. inserire i propri dati nella pagina di registrazione. 2. Cliccare su procedi 3. Cliccare su salva.
Risultati attesi	Deve apparire la pagina di lettura dei dati con i dati appena inseriti.

Test Case	**TC-009**
Nome	Pagina lettura dati, dati
Riferimento	REQ-005 Sub-REQ-002
Descrizione	Controllare che i dati vengano riportati correttamente.
Prerequisiti	Il prodotto deve essere completo
Procedura	1. inserire i propri dati nella pagina di registrazione. 2. Cliccare su procedi 3. Cliccare su salva.
Risultati attesi	Controllare che i dati mostrati siano quelli inseriti precedentemente.

Risultati test

Test Case	**Risultato**	**Descrizione**
TC-001	OK	-
TC-002	**Error**	I file vegnono scritti nella directory home del server, non dei file web.
TC-003	OK	-
TC-004	OK	-
TC-005	OK	-
TC-006	OK	-
TC-007	OK	-
TC-008	OK	-
TC-009	OK	-

TC-002

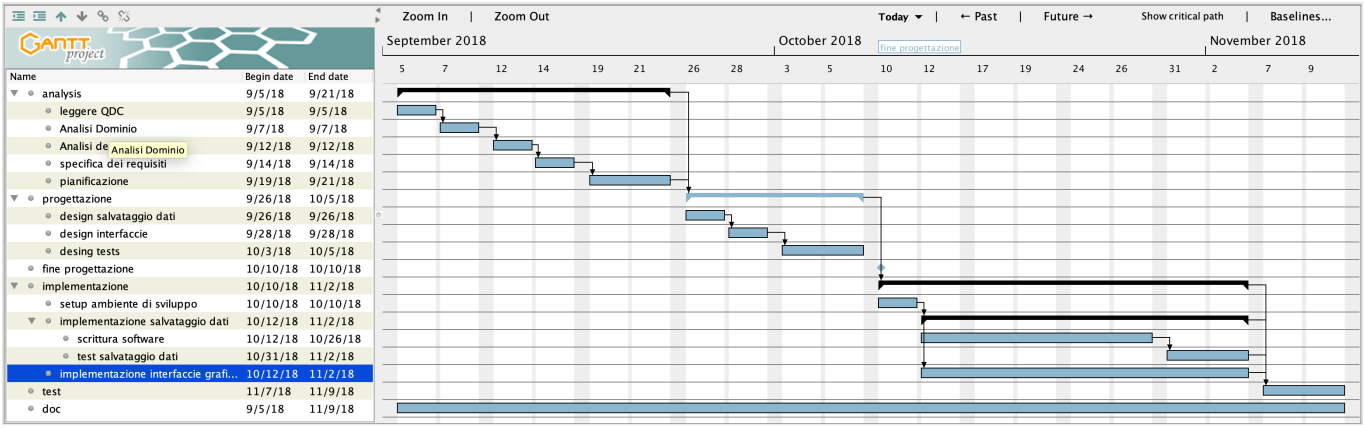
Si potrebbero creare delle servlet o delle pagine JSP che scrivano il file CSV. Il problema si riporta quando bisogna leggere il file con la data. Non c'è stato abbastanza tempo per implementare questa parte.

Mancanze/limitazioni conosciute

Nella pagina di lettura dei dati i dati non sono nella stessa posizione perché è stato scelto di poter mostrare anche gli altri dati. Quindi non vi era la possibilità di mantenere la stessa struttura della pagina.

Consuntivo

Per poter consegnare il prodotto in tempo e con le funzionalità di base ho dovuto lavorare anche fuori orario.



Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

Conclusioni

La soluzione che sono riuscito ad implementare è funzionale, e soprattutto è multipiattaforma. Aiuterà la società sportiva a raccogliere dati.

Mi è servito molto perché ho imparato come funzionano le basi di Apache Tomcat, il web server Java.

Sviluppi futuri

Si potrebbe creare una pagina di login, per le persone autorizzate a scaricare i file CSV.

Considerazioni personali

In questo progetto ho imparato a gestirmi meglio con i tempi di consegna, gestirmi il lavoro da eseguire ed in oltre ho imparato come funzionano le basi del web server Apache Tomcat per Java.

Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

Esempio:

- http://stackoverflow.com, *Stack Overflow*
- http://w3schools.com, *W3schools*
- http://tomcat.apache.org/, *Apache Tomcat*
- https://www.reddit.com/, *Reddit*
- https://angularjs.org/, *AngularJS*
- http://getbootstrap.com/, *Bootstrap*

Allegati

- Diari di lavoro