

Progetto1

Giulio Bosco - Progetto 1

1. Introduzione

- Informazioni sul progetto
- Abstract
- Scopo

2. Analisi

- Analisi del dominio
- Analisi e specifica dei requisiti
- Analisi dei mezzi
- Pianificazione

3. Progettazione

- Design dell'architettura del sistema
- Design dei dati e database
- Design delle interfacce
- Design procedurale

4. Implementazione

- Web Server
- Ambiente di sviluppo
- Sviluppo Applicativo
 - Front-end
 - Back-end

5. Test

- Protocollo di test
- Risultati test

6. Mancanze/limitazioni conosciute

7. Consuntivo

8. Conclusioni

- Sviluppi futuri
- Considerazioni personali

9. Sitografia

10. Allegati

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 su 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 <i>Introduzione</i>
Priorità	2
Versione	1.0
Note	pagina di benvenuto
Sotto requisiti	
001	Ci deve essere un tasto che porta alla pagina di inserimento dei dati.

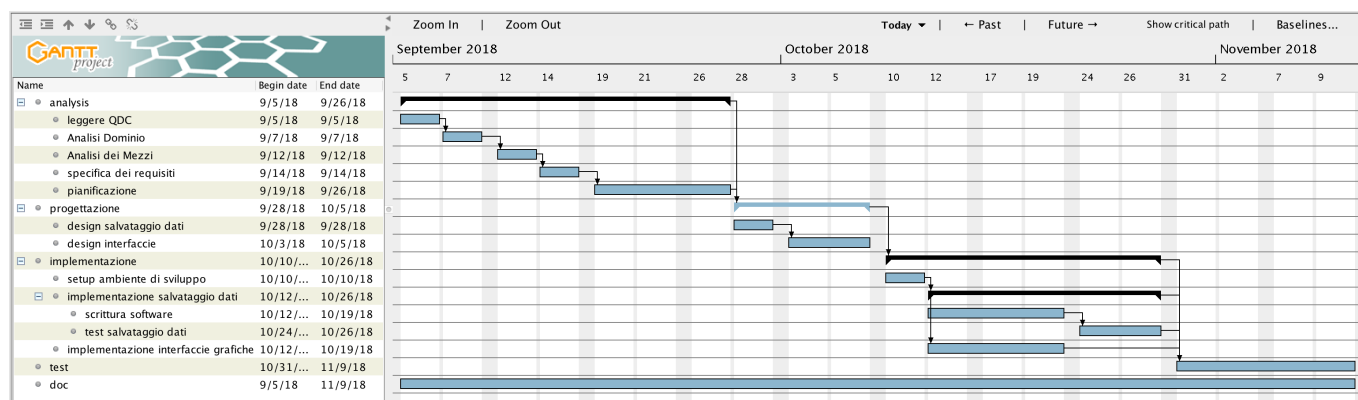
ID	REQ-004
Nome	Pagina <i>Inserzione Dati</i>
Priorità	1
Versione	1.0
Note	Form dei dati
Sotto requisiti	
001	Validazione dei dati
002	Ci deve essere un tasto che annulli l'inserimento fatto, cancelli tutti i contenuti dei campi.
003	Ci deve essere un tasto che porti alla pagina di controllo dei dati.

ID	REQ-005
Nome	Pagina <i>Controllo Dati</i>
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: <i>Inserzione Dati</i>
003	Ci deve essere un tasto per la registrazione, che scriva i dati sul csv e che poi porti alla pagina di <i>Lettura dei dati dal CSV</i>

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](#)

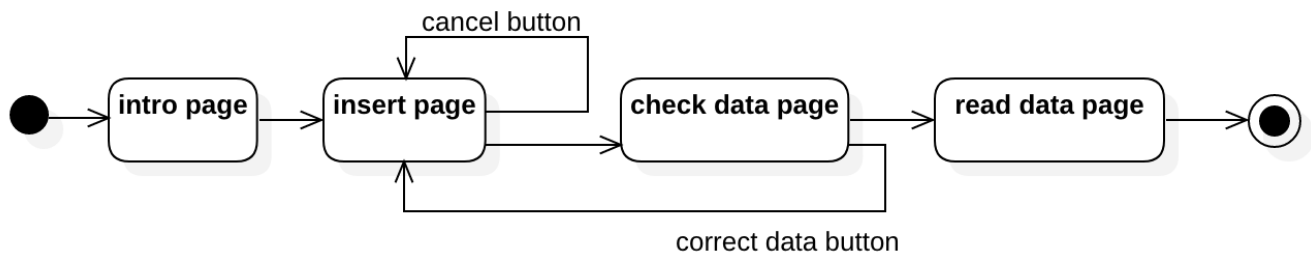
Hardware

- 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

Sport Club

insert your data
we will keep you updated

[enter your data](#)

In questa pagina vi è una introduzione al sito.

Con un bottone per spostarsi sulla pagina di inserimento dei dati.

Pagina inserimento dati

nome*	cognome*
data nascita*	
via *	numero civico *
città*	nap*
telefono*	
email*	
genere *	
hobby	
professione	
<input type="button" value="clear fields"/> <input type="button" value="submit"/>	

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

nome	cognome
data nascita	
via	numero civico
città	nap
telefono	
email	
genere	
hobby	
professione	

[edit data](#) [write data](#)

Nella pagina di controllo dei dati, vi saranno 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

nome cognome

data nascita

via numero civico

città nap

telefono

email

genere

hobby

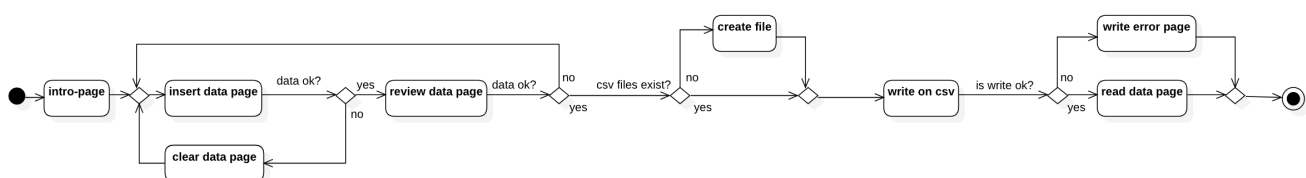
professione

back home

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 tre sezioni, con un menu sopra le quali.

Design procedurale



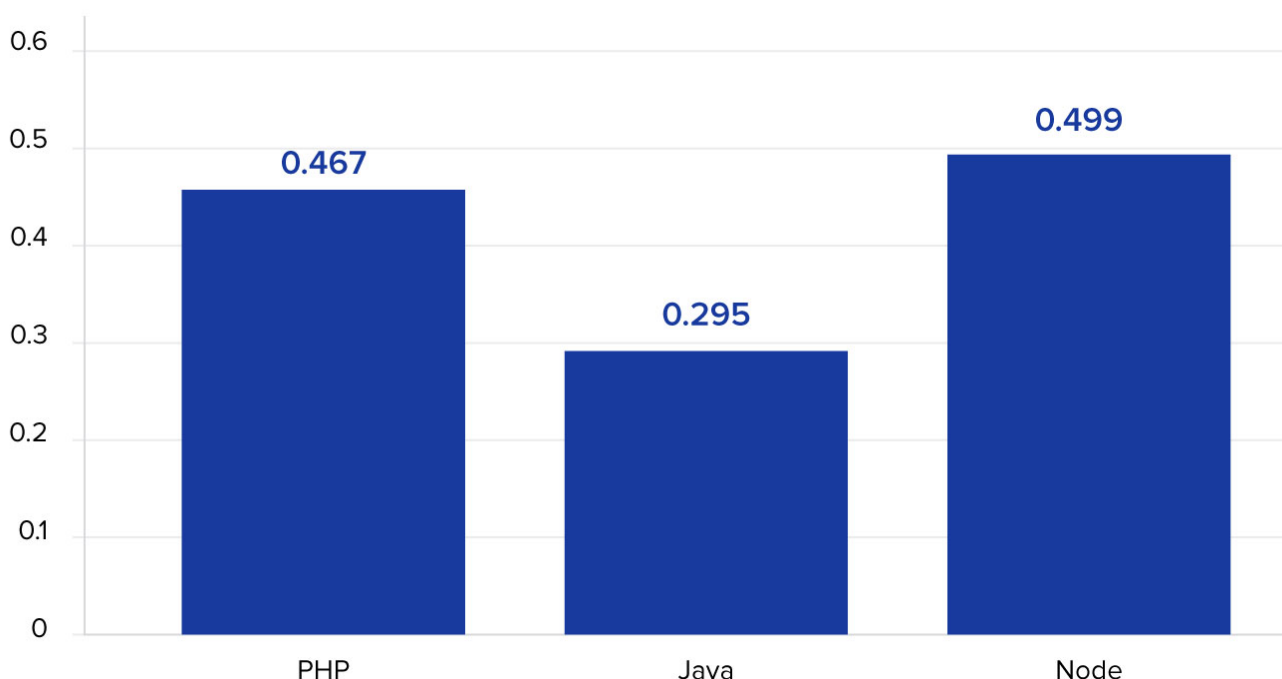
Entrando nel sito ci sarà una pagina di introduzione, poi la pagina di inserimento dati, dalla quale si può procedere oppure resettare tutti i campi. Proseguendo vi è la pagina di controllo dei dati, dalla quale si può procedere oppure ritornare sull'inserimento dei dati e modificare i nomi. Procedendo 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 opzioni di linguaggi di programmazione o scripting per eseguire le operazioni di lettura e scrittura sui file:

- PHP
- Java
- NodeJS



[toptal.com](https://www.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
```

Scaricare i file binari di Tomcat:

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

Per configurare le variabili d'ambiente di tomcat si può usare lo script fornito assieme ai file binari oppure manualmente (Guida per installazione manuale fornita assieme a tomcat). Lo script è disponibile sia per UNIX (**setenv.sh**), che per windows (**setenv.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**, lì 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 interfacce 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 opzioni, 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 domini 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 ai record del CSV.

- Csv - Gestione dei file CSV, inserimento e lettura dei dati
In questa classe vengono 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 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 è 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 Pages).

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	<ol style="list-style-type: none">1. inserire i propri dati nella pagina di registrazione.2. Cliccare su procedi3. 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	<ol style="list-style-type: none">1. inserire i propri dati nella pagina di registrazione.2. Cliccare su procedi3. 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	<p>Nelle due pagine aperte vi devono essere i file csv.</p> <p>Uno con tutte le registrazioni, mentre il secondo con le registrazioni giornaliere.</p> <p>All'interno dei csv i dati devono essere separati da un ;.</p>

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	<ol style="list-style-type: none">1. Controllare che sia stata 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	<ol style="list-style-type: none">1. inserire i propri dati nella pagina di registrazione.2. verificare che nessun campo venga sottolineato 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	<ol style="list-style-type: none">1. Inserire dei dati nei campi.2. Cliccare su annulla.
Risultati attesi	I campi devono essere vuoti.

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 vengono 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

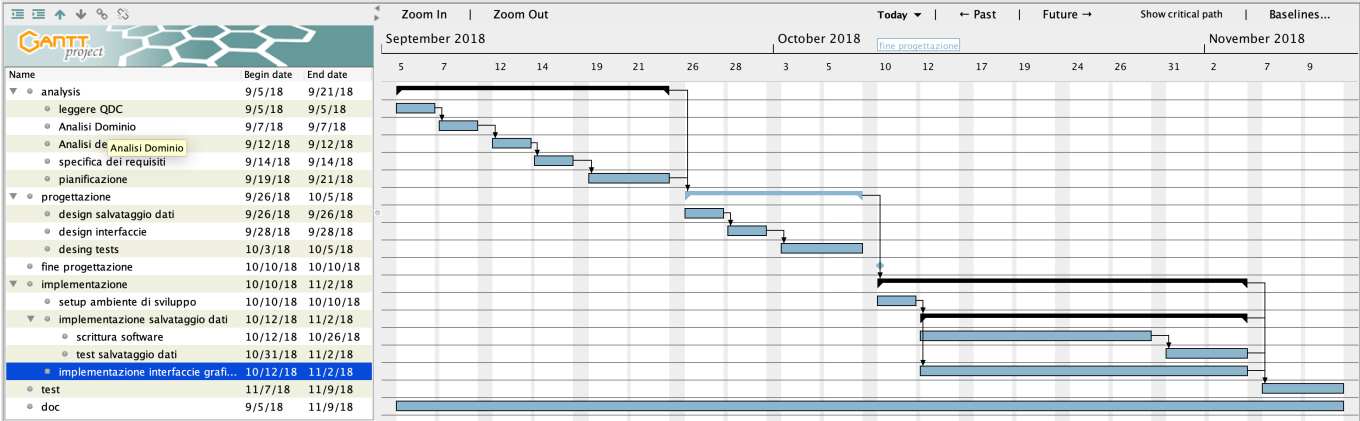
Si potrebbero creare delle servlet o delle pagine JSP che scrivano il file CSV. Il problema si riporta quando bisogna leggere il file con la data. Non c'è stato abbastanza tempo per implementare questa parte.

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 multiplatforma. 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
- Codice sorgente disponibile su: <https://github.com/boscogiulio/progetto1>

PROGETTO1 | Diario di lavoro - 05.09.2018

Giulio Bosco

Canobbio, 05.09.2018

Lavori svolti

Quello che faccio

Orario	Lavoro svolto
13:15 - 14:20	Spiegazione del lavoro
14:20 - 14:45	Consegna nuovo progetto

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Pianificare il lavoro sul progetto (GRANT)

PROGETTO | Diario di lavoro - 07.09.2018

Giulio Bosco

Canobbio, 07.09.2018

Orario	Lavoro svolto
13:15 - 14:45	Spiegazione documentazione
15:00 - 15:50	Teoria: Project lifecycle
15:50 - 16:15	Teoria: Project Analysis

Problemi riscontrati e soluzioni adottate

Nominazione generale dei file:

- diario: "YYYY.MM.DD_i3_Cognome_diary_prog1"
- doc: "YYYY.MM.DD_i3_Cognome_doc_prog1"

PROGETTO | Diario di lavoro - 12.09.2018

Giulio Bosco

Canobbio, 12.09.2018

Orario	Lavoro svolto
13:15 - 14:00	Domande a proposito del QDC
14:00 - 14:30	Inizio analisi del progetto
15:50 - 16:15	Risposta alle domande sul QDC

PROGETTO | Diario di lavoro - 14.09.2018

Giulio Bosco

Canobbio, 14.09.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 15:15	Analisi dei requisiti
15:05 - 16:30	Inizio pianificazione

Programma di massima per la prossima giornata di lavoro

finire la pianificazione

PROGETTO1 | Diario di lavoro - 19.09.2018

Giulio Bosco

Canobbio, 19.09.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 13:30	Pianificazione (Gantt)
13:30 - 13:45	Ricontrollato tabella dei requisiti con Muggiasca
13:45 - 14:45	Documentazione

PROGETTO | Diario di lavoro - 21.09.2018

Giulio Bosco

Lugano, 21.09.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:00	Pianificazione Gantt
14:00 - 14:45	Teoria
15:00 - 15:20	Diagramma di flusso
15:20 - 15:30	Diagramma delle pagine
15:30 - 15:45	Ricerca software mockup UI
15:45 - 16:20	Documentazione
16:20 - 16:30	Diario

Problemi riscontrati e soluzioni adottate

Diagrammi di flusso e/o procedurali

Software per disegnare i diagrammi di flusso e i diagrammi procedurali. \ Soluzione adottata: StarUML, perchè rispetta gli standard dei diagrammi UML, ha un utilizzo facile ed intuitivo.

Mockup UI

Software per disegnare le interfacce grafiche. \ Soluzione adottata: verrà utilizzato *Adobe PhotoShop*, perché si conosce già il software e non bisogna spendere tempo per imparare ad utilizzare altri software.

Programma di massima per la prossima giornata di lavoro

Durante la prossima lezioni vi sarà il test.

PROGETTO | Diario di lavoro - 26.09.2018

Giulio Bosco

Lugano, 26.09.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:30	Test
14:30 - 14:45	Pianificazione

PROGETTO | Diario di lavoro - 28.09.2018

Giulio Bosco

Lugano, 28.09.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Design delle interfacce
15:00 - 15:30	Diagramma dei dati
15:30 - 16:00	Documentazione
16:00 - 16:30	Diario

Problemi riscontrati e soluzioni adottate

Ho fatto il Design delle interfacce troppo dettagliato, quindi le ho rifatte, semplificandole. File finale: architecture/progetto1_views_desktop_2.psd

Programma di massima per la prossima giornata di lavoro

- iniziare la l'implementazione

PROGETTO | Diario di lavoro - 03.10.2018

Giulio Bosco

Lugano, 03.10.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:20	Teoria
15:20 - 15:30	Correzione Diagramma Gantt
14:30 - 14:45	Documentazione

Programma di massima per la prossima giornata di lavoro

- design dei tests

PROGETTO | Diario di lavoro - 05.10.2018

Giulio Bosco

Lugano, 05.10.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:00	Test Case 001
14:00 - 15:00	Configurare Tomcat WebServer
15:00 - 16:30	Code: Validators

Punto della situazione rispetto alla pianificazione

Leggermente in anticipo, già iniziata l'implementazione

Programma di massima per la prossima giornata di lavoro

Implementazione: - Creazione dei record - Gestione file CSV

PROGETTO | Diario di lavoro - 10.10.2018

Giulio Bosco

Canobbio, 10.10.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 13:30	Preparato ambiente di sviluppo
13:30 - 13:45	Implementato classe DateValidator (JavaClass)
13:45 - 14:15	Implementato classe Record (JavaClass)
14:15 - 14:30	Ripassare il funzionamento delle operazioni sui file in Java
14:30 - 14:45	Inizio implementazione classe Csv

Problemi riscontrati e soluzioni adottate

Ambiente di sviluppo

Come ambiente di sviluppo ho deciso di utilizzare

[JetBrains IntelliJ IDEA](#),

un tool professionale per lo sviluppo in Java.

Siccome la licenza "Education" è gratuita ho deciso di utilizzare questo software, che porta molte facilitazione durante lo sviluppo.

Scrittura in un file

Durante l'implementazione della class "Csv" ho avuto problemi con la scrittura nel file.

Soluzione:

Soluzione trovata su StackOverflow alla domanda How do I create a file and write to it in

Java?

[Link](#),

Consultazione: 10.10.2018

```
Files.write(filePath, lines, Charset.forName("UTF-8"));
```

Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione sono avanti, siccome ho impiegato solamente 15 minuti per configurare l'ambiente di sviluppo, quindi ho già iniziato a implementare.

Programma di massima per la prossima giornata di lavoro

Finire tutta la gestione dei Csv ed inizio dell'implementazione della grafica.

PROGETTO | Diario di lavoro - 12.10.2018

Giulio Bosco

Canobbio, 12.10.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 15:00	Implementato Classe Csv
15:00 - 15:15	Testato Classe Csv
15:15 - 16:15	Implementato Classe CsvToJson
16:15 - 16:30	Diario & Documentazione

Problemi riscontrati e soluzioni adottate

Classe Csv

Durante i test della classe Csv quando veniva creato il file il e aggiunto un record il record non aveva nessun separatore

Soluzione

Nel costruttore veniva settato il separatore solamente quando veniva aperto un file, settando il separatore come ultima operazione il problema viene risolto.

Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione sono avanti, pero non ho ancora iniziato a sviluppare la parte grafica.

Programma di massima per la prossima giornata di lavoro

Continuare Classe CsvToJson

PROGETTO1 | Diario di lavoro - 17.10.2018

Pinco Pallino

Canobbio, 17.10.2018

Lavori svolti

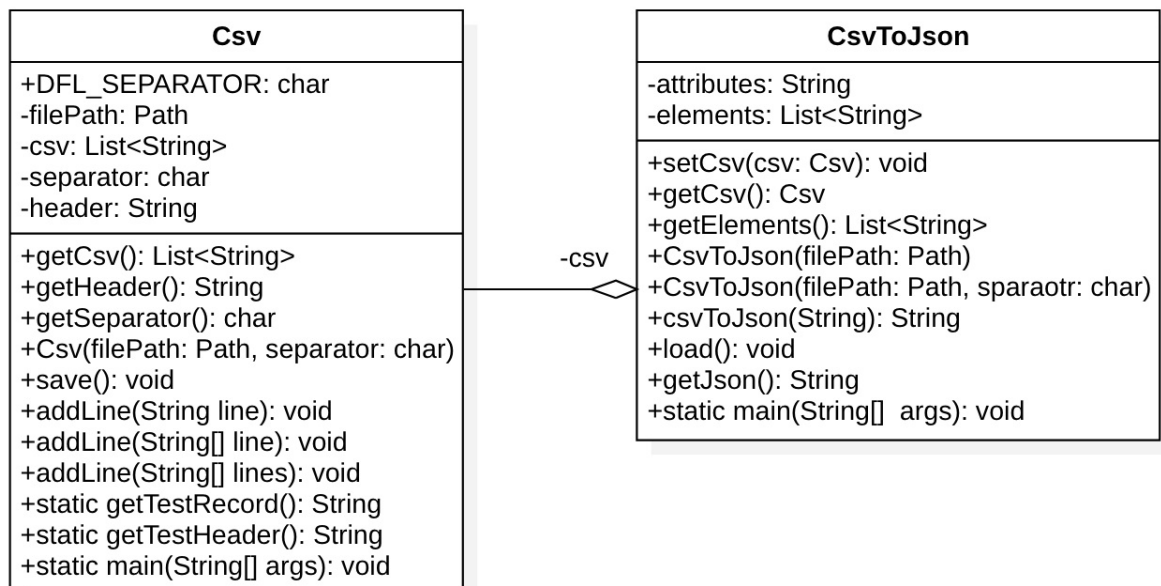
Orario	Lavoro svolto
13:15 - 13:50	Fine implementazione Classe CsvToJson
13:50 - 14:15	UML Diagrams
14:15 - 14:30	Diario
14:30 - 14:45	Documentazione

Fine implementazione Classe CsvToJson

La classe CsvToJson servirà per tradurre dei file CSV in dei file JSON.

I file CSV devono avere un header. L'output dei file JSON avrà questo stile.

```
[
{"attribute1":"value1", "attribute2": "value2"},
{"attribute1":"value1", "attribute2": "value2"}
]
```



Per usare la classe bisogna creare un istanza dell'oggetto con la path del file CSV e il separatore. Poi richiamare il metodo `Csv.getJson()` che ritorna una stringa con il contenuto del file JSON.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Sono Avanti rispetto alla pianificazione, ho già finito di implementare le classi per la scrittura dei dati.

Programma di massima per la prossima giornata di lavoro

Iniziare a sviluppare le interfacce grafiche.

PROGETTO1 | Diario di lavoro - 17.10.2018

Pinco Pallino

Canobbio, 17.10.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:10	Implementazione: struttura pagina "insert"
14:10 - 14:25	Implementazione: responsive pagina "insert"
14:25 - 14:45	Implementazione: data di nascita nella pagina "insert"
15:00 - 14:10	Implementazione: Record - getter per gli attributi del csv.
15:10 - 15:30	Implementazione: CSV - problema con setHeader()
15:30 - 16:00	Implementazione: Test della classe Record con scrittura sul CSV
16:00 - 16:05	Implementazione: aggiunto al NameValidator l'accettazione degli spazi
16:05 - 16:15	Diario
16:15 - 16:30	Implementazione: modificato metodo modifica Address.nap, Address.civicNumberLetter

Implementazione: struttura pagina "insert"

Desktop:

Sport Club - Insert Your Data

nome	cognome	
born date		
Indirizzo:		
Via	Numero Civico	
Città	NAP	
e-mail	Numero telefonico	
Genere: <input type="radio"/> Maschio <input type="radio"/> Femmina		
Hobby	Professione	
<input type="text"/>	<input type="text"/>	
<input type="button" value="Resetta valori"/>		<input type="button" value="Procedi>"/>

Mobile:

Sport Club Insert Your Data

nome	
cognome	
born date	
Indirizzo:	
Via	N Civico
Città	NAP
e-mail	
Numero telefonico	
Genere: <input type="radio"/> Maschio <input type="radio"/> Femmina	
Hobby	
<input type="text"/>	
Professione	
<input type="text"/>	
<input type="button" value="Resetta valori"/>	<input type="button" value="Procedi>"/>

Implementazione: Test della classe Record con scrittura sul CSV

Codice utilizzato per il test:

```
public class Test {  
    public static void main(String[] args) {  
        Record record = new Record();  
    }  
}
```

```

        record.setName("John");
        record.setSurname("Doe");
        record.setBornDate(new Date(2000,1,1));
        Address recordAddress = new Address();
        recordAddress.setStreet("Via Garibaldi");
        recordAddress.setCivicNumberLetter("57");
        recordAddress.setCity("milano");
        recordAddress.setNap("20121");
        record.setAddress(recordAddress);
        record.setEmail("john.doe@example.com");
        record.setMf('m');//record.setPhoneNumber(41796278810);
        record.setHobby("Play Football");
        record.setWork("Writer");

        try {
            Path path = Paths.get("Csv.txt");
            Csv writer = new Csv(path, ';');
            writer.setHeader(record.getAttributesStrings());

            writer.addLine(record.getDataStrings());

            writer.save();
        } catch (IOException ioe) {
            System.out.println("ioe");
        } catch (NoCsvHeaderException nche) {
            System.out.println("nche");
        }
    }
}

```

Problemi riscontrati e soluzioni adottate

Implementazione: struttura pagina "insert"

Radio buttons [solved]

Durante l'implementazione della struttura della pagina "insert" ho riscontrato dei problemi creando i radio button
potevano essere premuti entrambi.

Soluzione del problema trovata su: [w3schools.com - HTML](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_radio)
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_radio

Implementazione: CSV - problema con setHeader()

Il setter del header del csv con i parametri giusti veniva interrotto durante il controllo dei

parametri, l'ho risolto
corrgendo il controllo dei dati.

Punto della situazione rispetto alla pianificazione

Sono circa allo stesso livello della pianificazione

Programma di massima per la prossima giornata di lavoro

Sviluppare le interfacce grafiche.

PROGETTO1 | Diario di lavoro - 24.10.2018

Giulio Bosco

Canobbio, 24.10.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:15	Implementazione: Validators in JavaScript
14:15 - 14:20	Implementazione: Diario
14:20 - 14:45	Implementazione: Insert.js

Implementazione: Validators in JavaScript

ho implementato i seguenti validators:

- emailValidator
- localValidator
 - gender
 - phone number
 - nap
 - birth date
 - civic number
- nameValidator
- validator

Per creare i precedenti validators ho utilizzato StackOverflow per trovare le Regular Expression giuste:

- link 1: Name validator: <https://stackoverflow.com/questions/2013451/test-if-string-contains-only-letters-a-z-é-ü-ö-ê-å-ø-etc>
- link 2: <https://stackoverflow.com/questions/25726214/javascript-check-if-a-giving-string-contains-only-letters-or-digits>
- link 3: <https://stackoverflow.com/questions/46155/how-to-validate-an-email-address-in->

javascript

- link 4: numbers: <https://stackoverflow.com/questions/9011524/regex-to-check-whether-a-string-contains-only-numbers>
- link 5: dates: <https://stackoverflow.com/questions/7388001/javascript-regex-to-validate-date-format>

Problemi riscontrati e soluzioni adottate

Il nel insert.js non esegue correttamente la validazioni dei dati.

Punto della situazione rispetto alla pianificazione

Rispetto alla pianificazione sono abbastanza allineato,

Programma di massima per la prossima giornata di lavoro

Risolvere il problema della validazione front-end dei valori.

PROGETTO1 | Diario di lavoro - 26.10.2018

Giulio Bosco

Canobbio, 26.10.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 13:30	Implementazione: Classe Csv, Costruttore String path
13:30 - 13:40	Implementazione: Initialize Address
13:40 - 13:45	Aggiunto i diagrammi delle classi Csv e CsvToJson
13:45 - 14:00	Implementazione: Correzioni classi validator in JS
14:00 - 16:30	Implementazione: Cercare di capire perché viene scatenato un errore nelle servlet

Problemi riscontrati e soluzioni adottate

Errore nel salvataggio Csv dalle servlet.

Quando si esegue da linea di comando il `Csv.main()` non da problemi, crea il file se non esiste e inserisce i dati.

Mentre eseguendo lo stesso processo dalle servlet o dai file `.jsp` viene scatenata l'eccezione `NoCsvHeaderException` e sembra che non venga creato nessun file.

Dopo svariati tentativi non sono ancora riuscito a capire il perché.

Per cercare di capire dove vi è il problema ho provato ad eseguire il debugging dell'intero

codice, senza trovare l'errore.

Dopo scuola mi sono messo a provare ancora ed in poco tempo ho trovato qual'era il problema. Il file era già esistente ma vuoto nella cartella di Tomcat, quando si eseguono i programmi i file che vengono generati vengono scritti nella cartella su cui lavora Tomcat, cioè (/usr/local/tomcat/bin). In quella posizione vi era un file vuoto, quindi era giusto che venisse scatenata l'eccezione.

Punto della situazione rispetto alla pianificazione

Sono indietro rispetto alla pianificazione perché ho avuto problemi con le servlet di java.

PROGETTO1 | Diario di lavoro - 07.11.2018

Giulio Bosco

Canobbio, 07.11.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:00	Documentazione.
14:00 - 14:30	Raccolta material per la presentazione
14:30 - 14:45	Creazione struttura presentazione

Punto della situazione rispetto alla pianificazione

Sono avanti rispetto alla pianificazione, perché mi sono portato avanti fuori orario. Ho già completato i test e la documentazione.

Devo solamente finire la presentazione.