**Object Design**

1. **Introduzione**
   1. **Object design trade-offs**

[Costi vs Mantenimento]

Aggiungeremo della documentazione specifica per porzioni di codice avendo un aumento dei costi, ottenendo però in cambio una maggiore comprensibilità e maggiore facilità di manutenzione.

[Interfaccia vs Easy-use]

L'interfaccia verrà progettata in modo tale da essere intuitiva garantendo facilità d'uso anche ad utenti poco esperti. Il sistema userà font di grandezza adeguata per non appesantire la lettura e la navigazione sarà agevole. L'interfaccia si adatterà a qualsiasi dispositivo e manterrà le caratteristiche sopra descritte.

* 1. **Linee guida per l’interfaccia di documentazione**
* Le query di inserimento andranno su due righe: la prima conterrà la insert ed i parametri formali, la seconda i valori.
* Dopo un ciclo while o for seguirà la parentesi graffa, il blocco delle istruzioni verrà distanziato di un tab alla riga successiva.

Esempio:

while(true) {

Istruzioni

}

* Per la scrittura dei metodi, e per i tag html ci rifacciamo all’indentazione utilizzata per i cicli.

Esempio:

public metodo() {

istruzioni

}

Esempio:

<html>

<body>

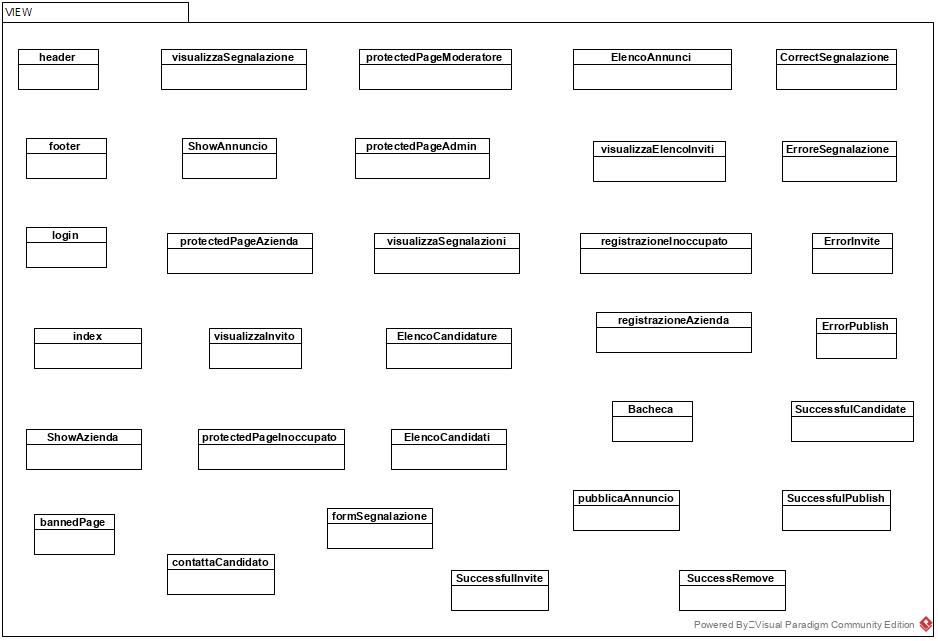
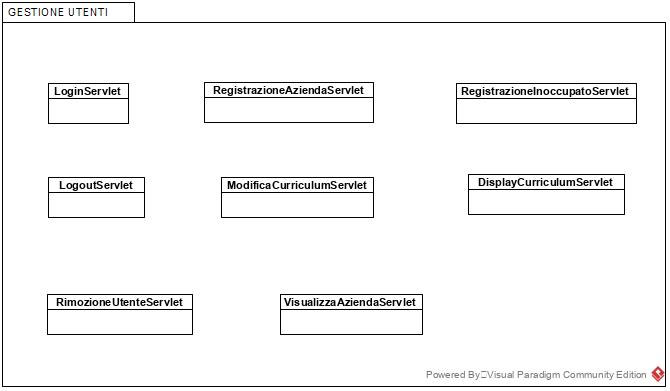
</body>

</html>

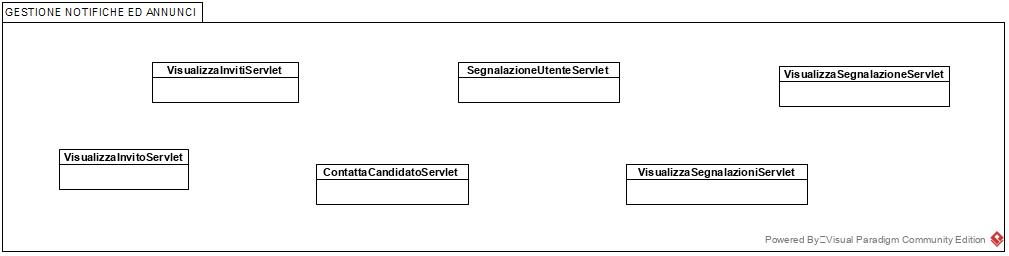
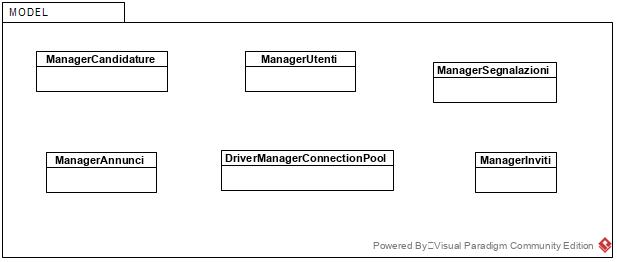
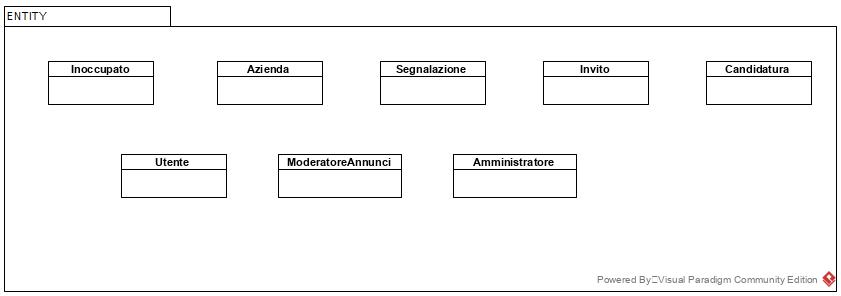
* Per le condizioni if con una sola istruzione utilizzeremo comunque le parentesi graffe.
* I nomi delle classi sono scritti al singolare e sono significativi rispetto al ruolo della classe.
* Anche i nomi delle variabili rispetteranno i parametri definiti per i nomi delle classi.
  1. **Componenti off-the-shelf**

Un componente che andremo ad utilizzare sarà Bootstrap, un Toolkit open source per lo sviluppo con HTML, CSS e JS. Risulta molto efficiente usare questo framework data la vasta gamma di librerie che mette a disposizione per la creazione dell’interfaccia utente.

* 1. **Definizioni, acronimi e abbreviazioni**
  + RAD: Requirements Analysis Document.
  + SDD: System Design Document.
  + ODD: Object Design Document.
  + DB: Database.
  + Off-the-shelf: componenti hardware/software disponibili sul mercato per l’acquisto da parte di aziende di sviluppo interessate ad utilizzarli.
  + Moderatore annunci: utente che si occupa della gestione degli annunci, della loro rimozione e della segnalazione degli utenti.
  + Amministratore: utente che si occupa della rimozione degli utenti e dell’analisi delle segnalazioni.
  + HTML: linguaggio di mark-up per pagine web.
  + Bootstrap: framework che fornisce un insieme di elementi grafici, stilistici, di impaginazione e di Javascript pronti all’uso.

1. **Packages**
   1. **Package Generale**
   2. **Package View**
   3. **Package Control** 
      1. **Package Gestione Utenti**
      2. **Package Gestione Bacheca**



* + 1. **Package Gestione Notifiche, Inviti E Segnalazioni**
  1. **Package Model**
  2. **Package Entity**

1. **Interfaccia delle classi**
   1. **ManagerCandidature**

|  |
| --- |
| **Nome Classe**: ManagerCandidature |
| **Descrizione Classe**: Classe composta da metodi statici che permettono di creare una candidatura e di visualizzare la lista delle candidature effettuate. |
| **Metodi:** + visualizzaCandidatureEffettuate(Inoccupato inocc)  + visualizzaCandidatureRicevute(int idAnn)  + candidate(int idInocc, int idAnn)  + isAlreadyCandidate(int idInoccupato, int idAnnuncio) |

|  |  |
| --- | --- |
| **Nome Metodo:** +visualizzaCandidatureEffettuate(Inoccupato inocc): List<Candidatura> | |
| **Descrizione metodo:** Dato un id di un inoccupato restituisce, se presente, una lista di candidature effettuate | |
| **PreCondizione** | **Context:** ManagerCandidature::visualizzaCandidatureEffettuate(Inoccupato inocc)  **pre**: (inoc != null) && (inoc.getId()>=1) |
| **PostCondizione** | **Context:** Context: ManagerCandidature::visualizzaCandidature (idInoccupato)  **post:**  // |

|  |  |
| --- | --- |
| **Nome Metodo:** +visualizzaCandidatureRicevute (int idAnn): List<Candidatura> | |
| **Descrizione metodo:** Dato un id di un annuncio restituisce, se presente, una lista di candidature ricevute per quell’annuncio. | |
| **PreCondizione** | **Context:** ManagerCandidature::visualizzaCandidatureRicevute(int idAnn)  **pre**: idAnn >= 1 |
| **PostCondizione** | **Context:** Context: ManagerCandidature::visualizzaCandidatureRicevute(int idAnn)  **post:**  // |

|  |  |
| --- | --- |
| **Nome Metodo:** +candidate(int idInocc, int idAnn): Boolean | |
| **Descrizione metodo:** Dato un id di un inoccupato e un id di un’annuncio, memorizza nel database la candidatura, solo se questo non si è già candidato per l’annuncio. | |
| **PreCondizione** | **Context:** CandidateManager::candidate(Inoccupato inocc,Annuncio ann)  **pre**: (inocc != null && inocc.getId()) >=1) && (ann != null && ann.getId()>=1) && !(isAlreadyCandidate (inocc.getId(), ann.getId()) |
| **PostCondizione** | **Context:** Context: CandidateManager::candidate(Inoccupato inocc,Annuncio ann)  **post:**  isAlreadyCandidate(inocc,getId(), ann.getId()) == true |

|  |  |
| --- | --- |
| **Nome Metodo:** +isAlreadyCandidate(int idInoccupato, int idAnnuncio): Boolean | |
| **Descrizione metodo:** Dato un id di un annuncio restituisce, se presente, una lista di candidature ricevute per quell’annuncio. | |
| **PreCondizione** | **Context:** ManagerCandidature::isAlreadyCandidate(int idInoccupato, int idAnnuncio)  **pre**: idAnnuncio >= 1 && idInoccupato >= 1 |
| **PostCondizione** | **Context:** Context: isAlreadyCandidate(int idInoccupato, int idAnnuncio)  **post:** Restituisce true se c’è già una candididatura con questi id, false altrimenti. |

* 1. **ManagerUtenti**

|  |
| --- |
| **Nome Classe**: ManagerUtenti |
| **Descrizione Classe**: Classe composta da metodi statici che permettono di: creare nuovi utenti, modificare informazioni su gli utenti e recupreare le loro informazioni |
| **Metodi:** + isPresent (Utente u)  + logIn(String username,string password)  + isAlreadyBanned (int idUser)  + deleteUser(Utente u)  + modificaCurriculum(int idUser, String pathNewCV)  + findAziendaById(int idUser)  + findInoccupato(int idUser)  + getNomeAzienda(ind idUser)  + registerUserInoccupato(Inoccupato inocc)  + registerUserAzienda(Azienda azienda)  - checkUser (String username, String nameTable)  - retriveUserInoccupato(String username,String password,String table)  - retriveUserAzienda(String username,String password,String table)  - retriveUserModeratore(String username,String password,String table)  - retriveUserAmministratore(String username,String password,String table) |

|  |  |
| --- | --- |
| **Nome Metodo:** +isPresent(Utente u): Boolean | |
| **Descrizione metodo:** Dato un utente si controlla che l’utente è presente o meno nel database. | |
| **PreCondizione** | **Context:** ManagerUtenti::isPresent(Utente u)  **pre**: (u != null) && (u.getUsername != “” && u.getUsername != null) |
| **PostCondizione** | **Context:** ManagerUtenti:: isPresent(Utente u)  **post:**  Restituisce true se :  checkUser(u.getUsername(),”Inoccupato”) OR  checkUser (u.getUsername(),”Azienda”) OR  checkUser (u.getUsername(),”Amministratore”)OR  checkUser(u.getUsername(),”Moderatore”) => TRUE  Altrimenti restituisce FALSE |

|  |  |
| --- | --- |
| **Nome Metodo:** +logIn(String username,String password): Utente u | |
| **Descrizione metodo:** Dato un utente si controlla che l’utente è presente o meno nel database, se è presente viene effettuato il login. | |
| **PreCondizione** | **Context:** ManagerUtenti::logIn(String username,String password)  **pre**: (username != “” && username != null) && (password != “” && password != null) |
| **PostCondizione** | **Context:** managerUtenti:: logIn(String username,String password)  **post:**  Restituisce un Inoccupato se retriveUserInoccupato(username,password,table) != null  Restituisce un Azienda se  retriveUserAzienda(username,password,table) != null  Restituisce un Moderatore se  retriveUserModeratore(username,password,table) != null  Restituisce un Amministratore se  RetriveUseAmministratore(username,password,table) != null  Altrimenti restituisce null, poiché l’username e la password non corrispondo a nessun utente nel database. |

|  |  |
| --- | --- |
| **Nome Metodo:** +isAlreadyBanned(int idUser): Boolean | |
| **Descrizione metodo:** Dato un id di un utente si controlla se l’utente è stato bannato. | |
| **PreCondizione** | **Context:** ManagerUtenti::idAlreadyBanned(String idUser)  **pre**: idUser >= 1 |
| **PostCondizione** | **Context:** ManagerUtenti:: isAlreadyBanned(String idUser)  **post:**  findAziendaById(idUser).getBanned() == true => TRUE  findAziendaById(idUser).getBanned() == tfalse => FALSE |

|  |  |
| --- | --- |
| **Nome Metodo:** +deleteUser(Utente u): Boolean | |
| **Descrizione metodo:** Dato un id di un utente questo viene rimosso, se presente dal database. | |
| **PreCondizione** | **Context:** ManagerUtenti::deleteUser(Utente u)  **pre**: (u.getId()>=1) && !(isAlreadyBanned(u.getId())) |
| **PostCondizione** | **Context:** ManagerUtenti:: deleteUser(Utente u)  **post:**  Restituisce true se è stato eliminarlo  Restituisce false se non è stato eliminato |

|  |  |
| --- | --- |
| **Nome Metodo:** +modificaCurriculum(int idUtente, String pathNewCV): Boolean | |
| **Descrizione metodo:** Dato un id di un utente questo viene rimosso, se presente dal database. | |
| **PreCondizione** | **Context:** ManagetUtenti::modificaCurriculum(int idUtente, String pathNewCV)  **pre**: idUtente>=1 && (String pathNewCV != “” && String pathNewCV != null) |
| **PostCondizione** | **Context:** ManagerUtenti::modificaCurriculum(int idUtente, String pathNewCV)  **post:**  findInoccupato(idUser).getCurriculum() == pathNewCV => TRUE  findInoccupato(idUser).getCurriculum() != pathNewCV => FALSE |

|  |  |
| --- | --- |
| **Nome Metodo:** +findAziendaById(int idUser): Azienda | |
| **Descrizione metodo:** Dato un id, restituisce l’azienda con l’id corrispondente. | |
| **PreCondizione** | **Context:** ManagerUtenti:: findAziendaById(int idUser)  **pre**: idUser != null && idUser >= 1 |
| **PostCondizione** | **Context:** ManagerUtenti:: findAziendaById(int idUser)  **post:** // |

|  |  |
| --- | --- |
| **Nome Metodo:** +findInocupato(int idUser): Inoccupato | |
| **Descrizione metodo:** Dato un id, restituisce l’inoccupato con l’id corrispondente. | |
| **PreCondizione** | **Context:** ManagerUtenti:: findInoccupato(int idUser)  **pre**: idUser != null && idUser >= 1 |
| **PostCondizione** | **Context:** ManagerUtenti:: findInoccupato(int idUser)  **post:**  // |

|  |  |
| --- | --- |
| **Nome Metodo:** +getNomeAzienda(int idUser): String | |
| **Descrizione metodo:** Dato un id, restituisce il nome dell’azienda con l’id corrispondente. | |
| **PreCondizione** | **Context:** ManagerUtenti:: findAziendaById(int idUser)  **pre**: idUser != null && idUser >= 1 |
| **PostCondizione** | **Context:** ManagerUtenti:: findAziendaById(int idUser)  **post:**  // |

|  |  |
| --- | --- |
| **Nome Metodo:** +registerUserInoccupato(Inoccupato inocc): Boolean | |
| **Descrizione metodo:** Dato un utente se questo non è già presente nel database, allora viene memoriazzato | |
| **PreCondizione** | **Context:** ManagerUtenti::registerUserInoccupato(Inoccupato inocc)  **pre**: !(isPresent(inocc)) |
| **PostCondizione** | **Context:** ManagerUtenti:: registerUserInoccupato(Inoccupato inocc)  **post:**  isPresent(inocc) == true |

|  |  |
| --- | --- |
| **Nome Metodo:** +registerUserAzienda(Inoccupato azienda): Boolean | |
| **Descrizione metodo:** Dato un utente se questo non è già presente nel database, allora viene memoriazzato | |
| **PreCondizione** | **Context:** ManagerUtenti::registerUserAzienda(Azienda azienda)  **pre**: !(isPresent(Azienda azienda)) |
| **PostCondizione** | **Context:** ManagerUtenti:: registerUserAzienda(Azienda azienda)  **post:**  isPresent(azienda) == true |

* 1. **ManagerInviti**

|  |
| --- |
| **Nome Classe:** ManagerInviti |
| **Descrizione Classe:** Classe che si occupa di gestire l’elenco di inviti di un inoccupato e la memorizzazione degli inviti creati da un’azienda per un inoccupato. |
| **Metodi:** + visualizzaInviti(Inoccupato inocc): List<Invito>  + contattaCandidato(Invito i): boolean  + getInvito(int idAnnuncio, int idInoccupato): Invito  - alreadyInvited(int idAnnuncio, int idInoccupato): boolean |

|  |  |
| --- | --- |
| **Nome Metodo:** +visualizzaInviti(Inoccupato inoccupato): List<Invito> | |
| **Descrizione metodo:** Dato un inoccupato, restituisce tutti gli inviti da lui ricevuti. | |
| **PreCondizione** | **Context:** ManagerInviti::visualizzaInviti(Inoccupato inocc)  **pre:** (inocc != null) && (inoccupato.getId() >= 1) |
| **PostCondizione** | **Context:** ManagerInviti::visualizzaInviti(Inoccupato inocc)  **post:** // |

|  |  |
| --- | --- |
| **Nome Metodo:** +contattaCandidato(Invito i): boolean | |
| **Descrizione metodo:** Dato un invito, lo rende persistente. | |
| **PreCondizione** | **Context:** ManagerInviti::contattaCandidato(Invito i)  **pre:** (i != null) && (i.getInoccupato().getId() >= 1) |
| **PostCondizione** | **Context:** ManagerInviti::contattaCandidato(Invito i)  **post:** getInvito(i.getAnnuncio(), i.getInoccupato()) != null && alreadyInvited(i.getAnnuncio(), i.getInoccupato()) == true => TRUE  getInvito(i.getAnnuncio(), i.getInoccupato()) == null && alreadyInvited(i.getAnnuncio(), i.getInoccupato()) == false => FALSE |

|  |  |
| --- | --- |
| **Nome Metodo:** +getInvito(int idAnnuncio, int idInoccupato): Invito | |
| **Descrizione metodo:** Restituisce un invito ricevuto dall’inoccupato con un determinato id per l’annuncio con un determinato id. | |
| **PreCondizione** | **Context:** ManagerInviti::getInvito(int idAnnuncio, int idInoccupato)  **pre:** idAnnuncio >= 1 && idInoccupato >= 1 |
| **PostCondizione** | **Context:** ManagerInviti::getInvito(int idAnnuncio, int idInoccupato)  **post:** // |

* 1. **ManagerSegnalazioni**

|  |
| --- |
| **Nome Classe:** ManagerSegnalazioni |
| **Descrizione Classe:** Classe che si occupa di gestire l’elenco di segnalazioni ricevute dall’amministratore e di gestire la memorizzazione delle segnalazioni create da un moderatore. |
| **Metodi:** + visualizzaElencoSengalazioni(): List<Segnalazione>  + segnalaUtente(Segnalazione segnalazione): boolean  + alreadyReported(int idMod, int idAz): boolean  + retrieveSingleReport(int idAz, int idMod): Segnalazione |

|  |  |
| --- | --- |
| **Nome Metodo:** +visualizzaElencoSegnalazioni(): List<Segnalazione> | |
| **Descrizione metodo:** Restituisce tutte le segnalazioni. | |
| **PreCondizione** | **Context:** ManagerSegnalazioni::visualizzaElencoSegnalazioni()  **pre:** // |
| **PostCondizione** | **Context:** ManagerSegnalazioni::visualizzaElencoSegnalazioni()  **post:** // |

|  |  |
| --- | --- |
| **Nome Metodo:** +segnalaUtente(Segnalazione s): boolean | |
| **Descrizione metodo:** Rende persistente una segnalazione. | |
| **PreCondizione** | **Context:** ManagerSegnalazioni::segnalaUtente(Segnalazione s)  **pre:** (segnalazione != null) && (s.getAzienda().getId >= 1) |
| **PostCondizione** | **Context:** ManagerSegnalazioni::segnalaUtente(Segnalazione s)  **post:** retrieveSingleReport(s.getAzienda(), s.getModeratore()) != null && areadyReported(s.getAzienda(), s.getModeratore) == true => TRUE  retrieveSingleReport(s.getAzienda(), s.getModeratore()) == null && areadyReported(s.getAzienda(), s.getModeratore) == false => FALSE |

|  |  |
| --- | --- |
| **Nome Metodo:** +alreadyReported(int idMod, int idAz): boolean | |
| **Descrizione metodo:** Controlla se l’azienda con l’id corrispondente ha già ricevuto una segnalazione dal moderatore con l’id corrispondete. | |
| **PreCondizione** | **Context:** ManagerSegnalazioni::alreadyReported(int idMod, int idAz)  **pre:** idMod >= 0 && idAz >= 0 |
| **PostCondizione** | **Context:** ManagerSegnalazioni::alreadyReported(int idMod, int idAz)  **post:** retrieveSingleReport(s.getAzienda(), s.getModeratore()) != null => TRUE  retrieveSingleReport(s.getAzienda(), s.getModeratore()) == null => FALSE |

|  |  |
| --- | --- |
| **Nome Metodo:** +retrieveSingleReport(int idAz, int idMod): Segnalazione | |
| **Descrizione metodo:** Restituisce la segnalazione fatta dal moderatore con l’id corrispondente all’azienda con l’id corrispondente. | |
| **PreCondizione** | **Context:** ManagerSegnalazioni::visualizzaElencoSegnalazioni()  **pre:** idAz >= 1 && idMod >= 1 |
| **PostCondizione** | **Context:** ManagerSegnalazioni::visualizzaElencoSegnalazioni()  **post:** // |

* 1. **ManagerAnnunci**

|  |
| --- |
| **Nome Classe:** ManagerAnnunci |
| **Descrizione Classe:** Classe statica che permette la ricerca, la visualizzazione, l’inserimento e la cancellazione degli annunci |
| **Metodi:** + searchAd(String ricerca): List<Annuncio>  + searchAdAdvanced(String tag, String città): List<Annuncio>  + visualizzaElencoAnnunci(Azienda azienda): List<Annuncio>  + pubblicaAnnuncio(Annuncio ann): boolean  + filterSearch(String data): List<Annuncio>  + removeAd(int idAnnuncio): boolean  + searchById(int idAnnuncio): Annuncio  - findTags(int idAnnuncio): List<String>  - inserimentoTag (int idAnnuncio,String tag): boolean  - deleteTag (int idAnnuncio): boolean |

|  |  |
| --- | --- |
| **Nome Metodo:** +searchAd(String ricerca): List<Annuncio> | |
| **Descrizione metodo:** Restituisce un elenco di annunci in base al tag specificato in input. | |
| **PreCondizione** | **Context:** ManagerAnnunci::searchAd(ricerca): List<Annuncio>  **pre:** ricerca != null |
| **PostCondizione** | **Context:** ManagerAnnunci::searchAd(ricerca): List<Annuncio>  **post:** // |

|  |  |
| --- | --- |
| **Nome Metodo:** +searchAdAdvanced(String tag, string città): List<Annuncio> | |
| **Descrizione metodo:** Restituisce un elenco di annunci in base alla città specificata in input. | |
| **PreCondizione** | **Context:** ManagerAnnunci::searchAdAdvanced(String tag, String città): List<Annuncio>  **pre:** città != null && !città.equals(“ ”) && tag != null && !tag.equals(“ ”) |
| **PostCondizione** | **Context:** ManagerAnnunci::searchAdAdvanced(String tag, String città): List<Annuncio>  **post:** // |

|  |  |
| --- | --- |
| **Nome Metodo:** +visualizzaElencoAnnunci(Azienda azienda): List<Annuncio> | |
| **Descrizione metodo:** Restituisce l’elenco di annunci pubblicati da un’azienda. | |
| **PreCondizione** | **Context:** ManagerAnnunci::visualizzaElencoAnnunci(azienda): List<Annuncio>  **pre:** (azienda != null) && (azienda.getId() >= 1) |
| **PostCondizione** | **Context:** ManagerAnnunci::visualizzaElencoAnnunci(azienda): List<Annuncio>  **post:** // |

|  |  |
| --- | --- |
| **Nome Metodo:** +pubblicaAnnuncio(Annuncio annuncio): boolean | |
| **Descrizione metodo:** Dato un annuncio, lo rende persistente. | |
| **PreCondizione** | **Context:** ManagerAnnunci::pubblicaAnnuncio(annuncio): boolean  **pre:** (annuncio != null) && (annuncio.getAzienda.getId() >= 1) |
| **PostCondizione** | **Context:** ManagerAnnunci::pubblicaAnnuncio(annuncio): boolean  **post:** searchById(annuncio.getIdAnnuncio()) != null |

|  |  |
| --- | --- |
| **Nome Metodo:** +filterSearch(String data): List<Annuncio> | |
| **Descrizione metodo:** Restituisce un elenco di annunci pubblicati nella data inserita in input. | |
| **PreCondizione** | **Context:** ManagerAnnunci::filterSearch(data): List<Annuncio>  **pre:** data <= Calendar.getInstance() |
| **PostCondizione** | **Context:** ManagerAnnunci::filterSearch(data): List<Annuncio>  **post:** // |

|  |  |
| --- | --- |
| **Nome Metodo:** +removeAd(int idAnnuncio): boolean | |
| **Descrizione metodo:** Rimuove un annuncio dal database. | |
| **PreCondizione** | **Context:** ManagerAnnunci::removeAd(int idAnnuncio): boolean  **pre:** idAnnuncio >= 1 |
| **PostCondizione** | **Context:** ManagerAnnunci::removeAd(int idAnnuncio): boolean  **post:** searchById(idAnnuncio)== null => true  searchById(idAnnuncio) != null => false |

|  |  |
| --- | --- |
| **Nome Metodo:** +seachById(int idAnnuncio): Annuncio | |
| **Descrizione metodo:** Restituisce un annuncio in base all’id passato come parametro | |
| **PreCondizione** | **Context:** ManagerAnnunci::searchById(idAnnuncio): Annuncio  **pre:** idAnnuncio != null && !(idAnnuncio.equals(“”) |
| **PostCondizione** | **Context:** ManagerAnnunci::searchById(idAnnuncio): Annuncio  **post:** // |

* 1. **RegistrazioneAziendaServlet**

|  |
| --- |
| **Nome Classe:** RegistrazioneAziendaServlet |
| **Descrizione Classe:** Servlet che si occupa di gestire la registrazione dell’azienda. Se tutti i dati sono stati inseriti correttamente e non ci sono omonimie, l’utente viene registrato e risulta loggato. |
| **Metodi:** +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doPost (HttpServletRequest request, HttpServletResponse response): void | |
| **Descrizione metodo:** Questo metodo si occupa di prendere i dati di input inseriti dall’azienda e memorizzarli al fine di renderla registrata al sito. | |
| **PreCondizione** | **Context:** RegistrazioneAziendaServlet :: doPost(request, response)  **pre:** (request.getParameter(“logoAzienda”) != null) && (request.getParameter(“logoAzienda”).substring(request.getParameter(“logoAzienda”) – 3) == “jpg” || request.getParameter(“logoAzienda”).substring(request.getParameter(“logoAzienda”) – 3) == “png”) &&  (request.getParameter(“nomeAzienda”) != null) && (formato nomeAzienda ^[A-Za-zàèìòù-9-.\_ ]{5,50}$) &&  (request.getParameter(“username”) != null) &&(formato username ^[A-Za-z0-9]{5,20}$) &&  (request.getParameter(“password”) != null) && (formato password ^[A-Za-z0-9-.\_]{8,16}$) &&  (request.getParameter(“confermaPasword”) != null) && (request.getParameter(“confermaPassword”).equals(request.getParameter(“password”))) &&  (request.getParameter(“email”) != null) && (formato email ^[A-Za-z0-9\_.][+@[a-zA-Z.]{2,}\.[a-zA-Z]{2,3}$](mailto:+@[a-zA-Z.]%7b2,%7d\.%5ba-zA-Z%5d%7b2,3%7d$)) &&  (request.getParameter(“indirizzoSede”) != null) && (formato indirizzoSede ^[A-Za-z ]{3,6}[A-Za-zàèìòù ]{2,35}[,]{1}[0-9 ]{2,5}$) && (request.getParameter(“dataFondazione”) != null) && (SimpleDateFormat("yyyy-MM-dd").parse(request.getParameter(“dataFondazione”)).before(new Date()) && (request.getParameter(“numeroDipendenti”) != null) && (Integer.parseInt(request.getParameter(“numeroDipendenti”)) >= 1) && request.getParameter(“partitaIVA”) != null) && (formato partitaIVA ^[A-Z0-9]{11,11}$) && (request.getParameter("trattamentoDati") != null) |
| **PostCondizione** | **Context:** RegistrazioneAziendaServlet :: doPost(request, response)  **Post:** // |

* 1. **VisualizzaAziendaServlet**

|  |
| --- |
| **Nome Classe:** VisualizzaAziendaServlet |
| **Descrizione Classe:** Questa classe è una servlet che si occupa di gestire la visualizzazione della  pagina di un’ azienda. |
| **Metodi** +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Effettua la visualizzazione della pagina di un’azienda. | |
| **Pre-Condizione** | **Context:** VisualizzaAziedaServlet::doGet(request, response)  **pre:** request.getParameter(“az”) >= 1 |
| **Post-Condizione** | **Context:** VisualizzaAziendaServlet::doGet(request,response)  **post:** response.getSession().getAttribute(“Azienda”) != null |

* 1. **VisualizzaCandidatureServlet**

|  |
| --- |
| **Nome Classe:** VisualizzaCandidatureServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire la visualizzazione delle  candidature di un inoccupato. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Effettua la visualizzazione della pagina con le candidature dell’inoccupato. | |
| **Pre-Condizione** | **Context:** VisualizzaCandidatureServlet::doGet(request, response)  **pre:** request.getSession().getAttribute("utenteInoccupato") != null |
| **Post-Condizione** | **Context:** VisualizzaCandidatureServlet::doGet(request,response)  **post:** response.getSession().getAttribute(“elencoAnn”) != null && request.getSession().getAttribute("candidature”) != null |

* 1. **VisualizzaCandidatiServlet**

|  |
| --- |
| **Nome Classe:** VisualizzaCandidatiServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire la visualizzazione dei  candidati per l’annuncio di un’azienda. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Effettua la visualizzazione della pagina l’elenco dei candidati. | |
| **Pre-Condizione** | **Context:** VisualizzaCandidatiServlet::doGet(request, response)  **pre:** Integer.parseInt(request.getParameter("idAn")) >= 1 && request.getParameter("tit") != null && !titoloAnnuncio.equals("") |
| **Post-Condizione** | **Context:** VisualizzaCandidatiServlet::doGet(request,response)  **post**: request.getSession().getAttribute("idAnnuncio”) != null && request.getSession().getAttribute("titoloAnnuncio”) != null && request.getSession().getAttribute("listaCandidati”) != null |

* 1. **DisplayCurriculumServlet**

|  |
| --- |
| **Nome Classe:** VisualizzaCurriculumServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire la visualizzazione del  curriculum di un candidato. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Effettua la visualizzazione del curriculum di un candidato. | |
| **Pre-Condizione** | **Context:** VisualizzaCurriculumServlet::doGet(request, response)  **pre:** session.getAttribute("utenteInoccupato") != null && session.getAttribute("utenteInoccupato") != null |
| **Post-Condizione** | **Context:** VisualizzaCurriculumServlet::doGet(request,response)  **post:** // |

* 1. **VisualizzaSegnalazioniServlet**

|  |
| --- |
| **Nome Classe:** VisualizzaSegnalazioniServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire la visualizzazione delle  segnalazioni effettuate da un utente. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Effettua la visualizzazione della pagina con la lista delle segnalazioni. | |
| **Pre-Condizione** | **Context:** VisualizzaSegnalzioniServlet::doGet(request, response)  **pre:**  request.getSession().getAttribute(“utenteAdmin”) != null |
| **Post-Condizione** | **Context:** VisualizzaCurriculumServlet::doGet(request,response)  **post:** response.getSession().getAttribute(“segnalazioni”) != null |

* 1. **RicercaAnnunciServlet**

|  |
| --- |
| **Nome Classe:** RicercaAnnunciServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire la ricerca degli annunci. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Si occupa di effettuare la ricerca di un annuncio utilizzando la stringa inserita dall’  Utente nella barra di ricerca. | |
| **Pre-Condizione** | **Context:** RicercaAnnunciServlet::doGet(request, response)  **pre:** request.getParameter("searchTag") != null && !(request.getParameter("searchTag")).equals(“ “) |
| **Post-Condizione** | **Context:** RicercaAnnunciServlet::doGet(request,response)  **post:** request.getSession().getAttribute("annunci") != null && request.getSession().getAttribute("tag”) != null && request.getSession().getAttribute("aziendeAnnunci") != null |

* 1. **PubblicaAnnuncioServlet**

|  |
| --- |
| **Nome Classe:** PubblicaAnnuncioServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire il form per la pubblicazione  di un annuncio. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Si occupa di effettuare la pubblicazione di un annuncio utilizzando il form  compilato dall’azienda. | |
| **Pre-Condizione** | **Context:** PubblicaAnnuncioServlet::doGet(request, response)  **pre:** request.getParameter(“titolo”)!=null &&  !(request.getParameter(“titolo”))equals(“ “) &&  formato titolo ^[A-Za-z0-9,. ]{6,50}$ &&  request.getParameter(“tags”) !=null &&  !(request.getParameter(“tags”)).equals(“ “) && formato tag ^[A-Za-z, ]{4,50}$ && request.getParameter(“descrizione”) !=null &&  !(request.getParameter(“descrizione”)).equals(“ “) &&  formato descrizione ^[A-Za-zàèìòù.,!?' ]{10,7000}$  request.getParameter(“requisiti”) ! = null &&  !(request.getParameter(“requisiti”)).equals(“ “) &&  formato requisiti ^[A-Za-zàèìòù0-9, ]{10,3000}$ && request.getParameter(“city”) ! = null &&  !(request.getParameter(“city”)).equals(“ “) &&  formato city ^[A-Za-z' ]{2,20}$ &&  request.getParameter(“tcontratto”) ! = null &&  solo un opzione tra:  - full-time  - part-time  - apprendistato  - progetto  - tirocinio formativo |
| **Post-Condizione** | **Context:** PubblicaAnnuncioServlet::doGet(request,response)  **post:** // |

* 1. **FiltraAnnunciServlet**

|  |
| --- |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire la ricerca avanzata di un  annuncio. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Si occupa di effettuare la ricerca avanzata di un annuncio filtrando per la città  Inserita dall’utente come parametro stringa. | |
| **Pre-Condizione** | **Context:** FiltraAnnunciServlet::doGet(request, response)  **pre:** request.getParameter(“advancedSearch”) != null && !request.getParameter(“advancedSearch”).equals(“ ”) && request.getParameter(“tag”) != null && !request.getParameter(“tag”).equals(“ ”) |
| **Post-Condizione** | **Context:** FiltraAnnunciServlet::doGet(request,response)  **post:** request.getSession().getAttribte(“annunci”) != null |

* 1. **ContattaCandidatoServlet**

|  |
| --- |
| **Nome Classe:** ContattaCandidatoServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire il form per l’invito al colloquio  con un candidato. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Si occupa di effettuare l’invio dell’invito utilizzando il form  compilato dall’azienda. | |
| **Pre-Condizione** | **Context:** ContattaCandidatoServlet::doGet(request, response)  **pre:** Integer.parseInt(request.getParameter("idAnnuncio")) >= 1 && Integer.parseInt(request.getParameter("idUtente")) >= 1 && request.getSession().getAttribute("utenteAzienda") != null &&  request.getParameter("titolo") != null && !request.getParameter("titolo").equals(“ “) && formato titolo ^[A-Za-zàèìòù .,!?']{5,60}$ && request.getParameter("messaggio") != null && !request.getParameter("messaggio").equals(“ “) && formato messaggio ^[A-Za-z\xE0\xE8\xEC\xF2\xF9 .,!?']{10,10000}$ |
| **Post-Condizione** | **Context:** ContattaCandidatoServlet::doGet(request,response)  **post:** // |

* 1. **SegnalazioneUtenteServlet**

|  |
| --- |
| **Nome Classe:** SegnalazioneUtenteServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire il form per l’invio di una  Segnalazione. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Si occupa di effettuare l’invio della segnalazione utilizzando il form inserito. | |
| **Pre-Condizione** | **Context:** SegnalazioneUtenteServlet::doGet(request, response)  **pre:** request.getParameter(“titolo”)!=null &&  !request.getParameter(“titolo”).equals(“ “) &&  Formato titolo [A-Z,a-z\é\è\ò\à\ù\ì\.,!?’] {5,60} &&  request.getParameter(“corpo”) !=null &&  !request.getParameter(“corpo”).equals(“ “) &&  formato Tag [A-Z,a-z\é\è\ò\à\ù\ì.,!?’] {10,10000} && Integer.*parseInt*(request.getParameter("azienda")) >= 1 && Integer.*parseInt*(request.getParameter("moderatore")) >= 1 |
| **Post-Condizione** | **Context**:: SegnalazioneUtenteServlet::doGet(request,response)  **post:** // |

* 1. **VisualizzaInvitiServlet**

|  |
| --- |
| **Nome Classe:** VisualizzaInviti |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire la visualizzazione degli inviti  per il colloquio |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Effettua la visualizzazione della pagina con l’elenco degli inviti. | |
| **Pre-Condizione** | **Context:** VisualizzaInviti::doGet(request, response)  **pre:** request.getSession().getParameter(“utente”) !=null |
| **Post-Condizione** | **Context:** VisualizzaInviti:: doGet(request,response)  **post:** response.getSession().getAttribute(“ListaInviti”)!=null |

* 1. **VisualizzaElencoAnnunciServlet**

|  |
| --- |
| **Nome Classe:** VisualizzaElencoAnnunciServlet |
| **Descrizione Classe** Questa classe è una servlet che si occupa di gestire la visualizzazione degli  Annunci pubblicati dall’azienda. |
| **Metodi**  +doGet(HttpServletRequest request, HttpServletResponse response):void  +doPost(HttpServletRequest request, HttpServletResponse response):void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response):void | |
| **Descrizione metodo** Effettua la visualizzazione della pagina con l’elenco degli annunci. | |
| **Pre-Condizione** | **Context:** VisualizzaElencoAnnunciServlet::doGet(request, response)  **pre:** request.getSession().getParameter(“utenteAzienda”) !=null |
| **Post-Condizione** | **Context:** VisualizzaElencoAnnunciServlet::doGet(request,response)  **post:** response.getSession().getAttribute(“listaAnnunci”)!=null |

* 1. **RimozioneUtenteServlet**

|  |
| --- |
| **Nome Classe:** RimozioneUtenteServlet |
| **Descrizione Classe:** Servlet che si occupa di gestire la rimozione di un utente. |
| **Metodi:** +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| **Nome Metodo**: +doPost (HttpServletRequest request, HttpServletResponse response): void | |
| **Descrizione metodo:** Questo metodo si occupa di rimuovere l’utente tramite id. | |
| **PreCondizione** | **Context:** RimozioneUtenteServlet :: doPost(request, response)  **pre**: Integer.parseInt(request.getParameter(“az”)) >= 1 |
| **PostCondizione** | **Context:** RimozioneUtenteServlet :: doPost(request, response)  **post**: getUserById(Integer.parseInt(request.getParameter(“az”))) == null |

* 1. **RimuoviAnnuncioServlet**

|  |
| --- |
| **Nome Classe:** RimuoviAnnuncioServlet |
| **Descrizione Classe:** Servlet che si occupa di gestire la rimozione di un annuncio |
| **Metodi:** +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet (HttpServletRequest request, HttpServletResponse response): void | |
| **Descrizione metodo:** Questo metodo si occupa di rimuovere l’annuncio tramite id. | |
| **PreCondizione** | **Context:** RimuoviAnnuncioServlet::doGet(request, response)  **pre:** Integer.parseInt(request.getParameter(“idDaRimuovere”)) >= 1 |
| **PostCondizione** | **Context:** RimozioneUtenteServlet :: doGet(request, response)  **post:** searchById(Integer.parseInt(request.getParameter(“idDaRimuovere”))) == null |

* 1. **CandidaturaServlet**

|  |
| --- |
| **Nome Classe:** CandidaturaServlet |
| **Descrizione Classe:** Servlet che si occupa di gestire le candidature effettuate dagli inoccupati. |
| **Metodi:** +doGet (HttpServletRequest request, HttpServletResponse response): void  +doPost (HttpServletRequest request, HttpServletResponse response): void |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet(HttpServletRequest request, HttpServletResponse response): void | |
| **Descrizione metodo:** Questo metodo si occupa di gestire la richiesta di candidatura di un inoccupato. | |
| **PreCondizione** | **Context:** CandidaturaServlet :: doGet(request, response)  **pre:** Integer.parseInt(request.getParameter(“idAz”)) >= 1 && Integer.parseInt(request.getParameter(“idUt”)) >= 1 |
| **PostCondizione** | **Context:** CandidaturaServlet :: doGet(request, response)  **post:** isAlreadyCandidate(Integer.parseInt(request.getParameter(“idUt”)), request.getParameter(“idAnnuncio”) == true |

* 1. **LoginServlet**

|  |
| --- |
| **Nome Classe**: LoginServlet |
| **Descrizione Classe**: Servlet che si occupa di gestire la login e di reindirizzare l’utente sulla pagina personale se questo è presente nel database, o su una pagina di errore in caso non sia presente nel database. |
| **Metodi:** +doGet (HttpServletRequest request, HttpServletResponse response)  +doPost (HttpServletRequest request, HttpServletResponse response) |

|  |  |
| --- | --- |
| **Nome Metodo:** +doPost (HttpServletRequest request, HttpServletResponse response): void | |
| **Descrizione metodo:** Questo metodo si occupa di far visualizzare i dati dell’utente prendendo dall’invio di una form, l’attributo username e l’attributo password | |
| **PreCondizione** | **Context:** LoginServlet::doPost (request,response)  **pre**: (request.getParameter(“username”) != null && !request.getParameter(“username”).equals(“ “))  &&  (request.getParameter(“password”) !=null && !request.getParameter(“password”).equals(“ “)) |
| **PostCondizione** | **Context:** Context: LoginServlet::doPost (request,response)  **post:** Se il login viene effettuato da un utente azienda: response.getSession().getAttribute(“utenteAzienda”) != null  Se il login viene effettuato da un utente inoccupato:  response.getSession().getAttribute(“utenteInoccupato”) != null  Se il login viene effettuato da un utente moderatore annunci:  response.getSession().getAttribute(“utenteModeratore”) != null  Se il login viene effettuato da un utente amministratore:  response.getSession().getAttribute(“utenteAdmin”) != null |

* 1. **ModificaCurriculumServlet**

|  |
| --- |
| **Nome Classe**: ModificaCurriculumServlet |
| **Descrizione Classe**: Servlet che si occupa di gestire la modifica del curriculum per gli utenti inoccupati |
| **Metodi:** +doGet (HttpServletRequest request, HttpServletResponse response)  +doPost (HttpServletRequest request, HttpServletResponse response) |

|  |  |
| --- | --- |
| **Nome Metodo:** +doPost (HttpServletRequest request, HttpServletResponse response): void | |
| **Descrizione metodo:** Questo metodo si occupa di fornire all’inoccupato la possibilità di modificare il proprio cv. | |
| **PreCondizione** | **Context:** ModificaCurriculumServlet::doPost (request,response)  **pre**: request.getPart("curriculum") != null && request.getPart("curriculum").getSize() != 0 && formato curriculum .pdf && formato curriculum max 10MB &&  session.getAttribute("utenteInoccupato") != null |
| **PostCondizione** | **Context:** Context: ModificaCurricuulmServlet::doPost (request,response)  **post:** request.getSession().getAttribute(“utente”).getCV () == request.getParameter(“pdf”) |

* 1. **LeggiAnnuncioServlet**

|  |
| --- |
| **Nome Classe**: LeggiAnnunciServlet |
| **Descrizione Classe**: Servlet che si occupa di far visualizzare un annuncio all’inoccupato che lo clicca |
| **Metodi:** +doGet (HttpServletRequest request, HttpServletResponse response)  +doPost (HttpServletRequest request, HttpServletResponse response) |

|  |  |
| --- | --- |
| **Nome Metodo:** +doGet (HttpServletRequest request, HttpServletResponse response): void | |
| **Descrizione metodo:** Questo metodo si occupa di fornire all’inoccupato la possibilità di visualizzare l’annuncio su cui ha cliccato | |
| **PreCondizione** | **Context:** LeggiAnnunciServlet::doGet (request,response)  **pre**: Integer.*parseInt*(request.getParameter("idAnnuncio")) >= 1 |
| **PostCondizione** | **Context:** LeggiAnnunciServlet::doGet (request,response)  **post:** request.getSession().getAttribute(“annuncioSelezionato”) != null |

* 1. **Utente**

|  |
| --- |
| **Utente** |
| * idUser: int * Username: String * Password: String * Email: String |
| +getIdUser(): int  +getUsername(): String  +getPassword (): String  +getEmail(): String  +setIdUser(int id): void  +setUsername(String username): void  +setPassword (Inoccupato password): void  +setEmail(String email): void |

* 1. **Inoccupato**

|  |
| --- |
| **Inoccupato** |
| * idUser: int * Username: String * Password: String * Email: String * Nome: String * Cognome: String * Data di nascita: Date * Residenza: String * CV: String |
| +getIdUser(): int  +getUsername(): String  +getPassword (): String  +getEmail(): String  +getNome(): String  +getCognome(): String  +getNascita(): Date  +getResidenza(): String  +getCV(): String  +setIdUser(int id): void  +setUsername(String username): void  +setPassword (Inoccupato password): void  +setEmail(String email): void  +setNome(String nome): void  +setCognome(String cognome): void  +setNascita(Date nascita): void  +setResidenza(String residenza): void  +setCV(String cv): void |

* 1. **Azienda**

|  |
| --- |
| **Azienda** |
| * idUser: int * Username: String * Password: String * Email: String * Nome azienda: String * Numero dipendenti: int * Logo azienda: String * Indirizzo sede: String * PartitaIVA: String * Descrizione: String * Banned: boolean |
| +getIdUser(): int  +getUsername(): String  +getPassword (): String  +getEmail(): String  +getNomeAzienda(): String  +getNumDipendenti():int  +getIndirizzo (): String  +getLogoAzienda(): String  +getPartitaIVA(): String  +getDescrizione (): String  +getBanned(): boolean  +setIdUser(int id): void  +setUsername(String username): void  +setPassword (Inoccupato password): void  +setEmail(String email): void  +setIndirizzo(String nuovoIndirizzo): void  +setNomeAzienda(String nomeAzienda): void  +setLogoAzienda(String logoAzienda): void  +setNumDipendenti(int newNumber): void  +setPartitaIVA(String partitaIVA): void  +setDescrizione (String descrizione): void  +setBanned(boolean banned): void |

* 1. **Moderatore**

|  |
| --- |
| **Moderatore** |
| * idUser: int * Username: String * Password: String * Email: String |
| +getIdUser(): int  +getUsername(): String  +getPassword (): String  +getEmail(): String  +setIdUser(int id): void  +setUsername(String username): void  +setPassword (Inoccupato password): void  +setEmail(String email): void |

* 1. **Amministratore**

|  |
| --- |
| **Amministratore** |
| * idUser: int * Username: String * Password: String * Email: String |
| +getIdUser(): int  +getUsername(): String  +getPassword (): String  +getEmail(): String  +setIdUser(int id): void  +setUsername(String username): void  +setPassword (Inoccupato password): void  +setEmail(String email): void |

* 1. **Annuncio**

|  |
| --- |
| **Annuncio** |
| * idAnnuncio: int * Titolo: String * Descrizione: String * Requisiti: String * Tags: List<String> * Azienda: int * NomeAzienda: String * DataPubblicazione: String * Tipo contratto: String * Città : String |
| +getIdAnnuncio(): int  +getTitolo(): String  +getDescrizione(): String  +getRequisiti(): String  +getTags(): List<String>  +getAzienda(): Azienda  +getNomeAzienda(): int  +getData(): String  +getTipoContratto(): String  +getCittà():String  +setIdAnnuncio(int id): void  +setTitolo(String titolo): void  +setDescrizione(String descrizione): void  +setRequisiti(String requisiti): void  +setTags(List<String> tags): void  +setAzienda(int azienda): void  +setNomeAzienda(String nomeAzienda): void  +setData(String data): void  +setTipoContratto(String contratto): void  +setCittà(String città): void |

* 1. **Candidatura**

|  |
| --- |
| **Candidatura** |
| * Inoccupato: int * Azienda: int * Data: String |
| +getData(): Date  +getAzienda(): int  +getInoccupato(): int  +setData(String data): void  +setAzienda(int azienda): void  +setInoccupato(int inoccupato): void |

* 1. **Invito**

|  |
| --- |
| **Invito** |
| * Inoccupato: int * Azienda: int * Annuncio: int * NomeAzienda: String * Titolo: String * Corpo: String |
| +getAzienda(): int  +getInoccupato(): int  +getAnnuncio(): int  +getNomeAzienda(): String  +getTitolo(): String  +getCorpo(): String  +setAzienda(int azienda): void  +setInoccupato(int inoccupato): void  +setAnnuncio(int annuncio): void  +setNomeAzienda(String nomeAzienda): void  +setTitolo(String titolo): void  +setCorpo(String corpo): void |

* 1. **Segnalazione**

|  |
| --- |
| **Segnalazione** |
| * Titolo: String * Corpo: String * Azienda: int * Data: int * Moderatore: int |
| +getData(): String  +getAzienda(): int  +getTitolo (): String  +getCorpo(): String  +getModeratore(): int  +setData(String data): void  +setAzienda(int azienda): void  +setTitolo(String titolo): void  +setCorpo (String corpo): void  +setModeratore(int moderatore): void |

* 1. **DriverConnectionPool**

|  |
| --- |
| DriverConnectionPool |
| -Static List: List<Connection> freeDbConnections |
| -static synchronized createDBConnection(): Connection  +static synchronized getConnection(): Connection  + static synchronized releaseConnection(Connectionc c): void |

1. **Class Diagram Finale**

