****

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **O**bject **D**esign  **D**ocument  Easy Pass   |  |  | | --- | --- | | Riferimento |  | | Versione | 1.0 | | Data | 9/12/2021 | | Destinatario | Prof. C. Gravino | | Presentato da | Montefusco Alberto  Mulino Martina  Rinaldi Viviana  Spina Gennaro | | Approvato da |  | |

Sommario

[Sommario 2](#_Toc530825397)

[RevisionHistory 3](#_Toc530825396)

[Responsabilità 4](#_Toc530825396)

1. [Introduzione 5](#_Toc530825398)

1.1 Object Design Trade-off 5

1.2 Componenti off-the-shelf 5

1.3 Linee guida per la documentazione di Interfacce ........................................................................... 7

1.4 Design Pattern ...................................................................................................................................... 7

1.5 Definizioni, acronimi e abbreviazioni ............................................................................................... 8

1.6 Riferimenti ............................................................................................................................................ 9

1. Packages ........................................................................................................................................................... 9

2.1 Interface ................................................................................................................................................ 9

2.2 Application Logic ................................................................................................................................. 9

2.3 Storage ................................................................................................................................................... 9

1. Class Interfaces ................................................................................................................................................ 9
2. Class Diagram................................................................................................................................................ 18
3. Glossario ........................................................................................................................................................ 20

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 13/12/2021 | 0.1 | Prima stesura: divisione dei compiti | Alberto Montefusco  Gennaro Spina  Viviana Rinaldi  Martina Mulino |
| 13/12/2021 | 0.2 | Aggiunta dei trade-off e delle linee guida per la documentazione delle interfacce | Alberto Montefusco  Viviana Rinaldi |
| 14/12/2021 | 0.3 | Aggiunta dei package del progetto | Martina Mulino |
| 15/12/2021 | 0.4 | Aggiunta dei Design Pattern | Alberto Montefusco  Gennaro Spina  Viviana Rinaldi  Martina Mulino |

Responsabilità

|  |  |
| --- | --- |
| **Artefatto** | **Autori** |
| Object Design  Trade-off | Viviana Rinaldi |
| Components  off-the-shelf | Gennaro Spina |
| Linee guida per la documentazione delle interfacce | Alberto Montefusco |
| Package | Martina Mulino |
| Design Pattern | Alberto Montefusco, Viviana Rinaldi, Martina Mulino, Gennaro Spina |
| Class Interfaces: Esito, EsitoDAO, Dipartimento, DipartimentoDAO | Alberto Montefusco |
| Class Interfaces: DirettoreDiDipartimento, DirettoreDiDipartimentoDAO, Docente, DocenteDAO | Martina Mulino |
| Class Interfaces: Report, ReportDAO, SessioneDiValidazione, SessioneDiValidazioneDAO | Gennaro Spina |
| Class Interfaces: Formato, FormatoDAO, SessionController, AccessController, ReportController | Viviana Rinaldi |

1. Introduzione

L’Object Design Document illustra i diversi dettagli legati alla fase implementativa del Sistema Easy Pass; in particolare, esso descrive i trade-off di progettazione definiti dagli sviluppatori, le linee guida da seguire per le interfacce dei sottosistemi, i design pattern utilizzati, la decomposizione dei sottosistemi in packages e classi e, infine, la specifica delle interfacce delle classi.

* 1. Object Design Trade-off

**Tempo di rilascio Vs funzionalità**

Per rispettare le scadenze del Progetto potrebbe essere necessaria l’implementazione parziale di alcune delle funzionalità richieste.

**Portabilità Vs Efficienza**

Il Sistema è progettato per far in modo che Browser diversi e dispositivi aventi risoluzioni diverse (come smartphone, laptop o computer desktop) possano visualizzare correttamente le pagine web del sito Easy Pass sfruttando al meglio lo spazio del display. Questo implica che il livello di efficienza garantito non sia lo stesso per ogni dispositivo, poiché una tale adattabilità richiederebbe un carico maggiore da gestire.

**Velocità Vs memoria**

Per garantire tempi di risposta rapidi, si è preferito utilizzare query che risultano più veloci a discapito dello spazio che occupano in memoria; in particolar modo viene introdotta ulteriore ridondanza di dati.

**Costruire Vs Comprare**

Sebbene utilizzare software già realizzato da altri permetta, ad esempio, l’utilizzo di funzionalità già complete oppure una minore quantità di lavoro per gli sviluppatori, è stato deciso di realizzare la maggior parte del Sistema partendo da zero, utilizzando componenti esterne soltanto in alcuni casi (es. integrazione di librerie open source fornite dal Ministero della Salute per la validazione dei Green Pass). Il motivo per cui è stata presa questa decisione riguarda l’aumento dei costi e l’impegno necessario per integrare le componenti già realizzate con quelle costruite dagli sviluppatori.

Nella seguente tabella, il Design Goal in **grassetto** indica il design goal prioritario.

|  |  |
| --- | --- |
| **Trade-Off** | |
| **Tempi di rilascio** | Funzionalità |
| **Portabilità** | Efficienza |
| **Velocità** | Memoria |
| **Costruire** | Comprare |

* 1. Componenti off-the-shelf

Il Sistema utilizzerà i seguenti componenti off-the-shelf:

* **Apache Tomcat**, un web server con annesso application container per applicazioni scritte in Java;
* **Node.js**, un runtime system open source multipiattaforma orientato agli eventi per l'esecuzione di codice JavaScript;
* **jsQR**, una libreria JavaScript per la letture di codici QR da immagini
* **dcc-utils**, un Package NPM (Node Package Manager) che contiene una serie di tool per leggere e validare i Green Pass;
  1. Linee guida per la documentazione di Interfacce

Il Sistema Easy Pass è realizzato con l’IDE di sviluppo IntelliJ IDEA 2021.3 ed è strutturato nel seguente modo:

* Il progetto è suddiviso in tre package principali (Interface, Storage, Application Logic) i quali contengono i rispettivi sub-package.
* Il nome di una classe deve rispettare il seguente formato: **Nomeclasse**.
* Il nome di un metodo o di una variabile di istanza deve rispettare la notazione camel case.
* Un intero metodo, compreso di intestazione ed istruzioni, è preceduto e seguito da una riga vuota.
* I commenti, laddove necessari, avranno formato **// commento** se si estendono su una sola riga, altrimenti se un commento si estende su più righe presenta il formato **/\* commento \*/**.
* Ogni classe e ogni metodo devono essere corredate da commenti che rispettano lo standard utilizzato da Javadoc per la produzione di documentazione in formato HTML.
* Il package Storage contiene dei sub-package in cui sono presenti tutte le classi che fanno riferimento ad entità persistenti (Bean e DAO).
* Ogni classe contenuta nel package Storage deve contenere un costruttore e i metodi getter e setter.
* Il package Application Logic contiene tutte le classi Servlet (@WebService) che si occupano della logica di business del Sistema e agisce da interlocutore tra le classi contenute nei sub-package del package Storage e Interface.
* Il package Interface contiene dei sub-package in cui sono organizzati tutti i file che si occupano dell’interfaccia utente (JSP, pagine HTML).
* I file CSS si trovano nel package WEB-INF del progetto e in particolare in un sub-package chiamato “css”.
* I file JavaScript si trovano nel package WEB-INF del progetto e in particolare in un sub-package chiamato “js”.
* Le dichiarazioni si trovano sempre all’inizio dei corrispondenti blocchi di codice, poiché dichiarare le variabili soltanto in corrispondenza del loro primo utilizzo può provocare confusione.
  1. Design Pattern

DAO

Singleton

Adapter Pattern ???

* 1. Definizioni, acronimi e abbreviazioni
  2. Riferimenti

Per stilare la presente documentazione, si è preso come riferimento le slide fornite dal Docente del corso di Ingegneria del Software, Carmine Gravino, inserite nella sezione “M4” della piattaforma di e-learning della facoltà di Informatica. Inoltre, è stato consultato il libro di testo “Object-Oriented Software Engineering Using UML, Patterns and Java: Third Edition, di Bernd Bruegge ed Allen H. Dutoit”.

1. Packages
   1. Interface

Questo package contiene i seguenti sub-package e le seguenti classi:

* Package StudenteGUI:
  + SendGP
* Package DocenteGUI:
  + AvviaSessione
  + ElencoEsiti
  + AnteprimaReport
  + Registrazione
* Package DirettoreDiDipartimentoGUI:
  + HomePage
  + GestioneReport
  + GestioneFormato
* Package AutenticazioneGUI
  + Login
  1. Application Logic

Questo package contiene le seguenti classi:

* SessionController
* AccessController
* ReportController
  1. Storage

Questo package contiene i seguenti sub-package e le seguenti classi:

* Package DirettoreDiDipartimento:
  + DirettoreDiDipartimento
  + DirettoreDiDipartimentoDAO
* Package Docente:
  + Docente
  + DocenteDAO
* Package Esito:
  + Esito
  + EsitoDAO
* Package Dipartimento:
  + Dipartimento
  + DipartimentoDAO
* Package Report:
  + Report
  + ReportDAO
* Package SessioneDiValidazione:
  + SessioneDiValidazione
  + SessioneDiValidazioneDAO
* Package Formato:
  + Formato
  + FormatoDAO

1. Class Interfaces

Di seguito, vengono elencate tutte le classi previste dal Sistema, in particolare dei package Application Logic e Storage.

|  |  |
| --- | --- |
| **Nome Classe** | **Esito** |
| **Descrizione** | Questa classe è un semplice POJO che modella degli oggetti Esito. |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | **EsitoDAO** |
| **Descrizione** | Questa classe contiene i metodi che permettono di effettuare operazioni CRUD per un Esito. |
| **Pre-condizione** | **context** Esito :: findEsitoById(id)  **pre**: findEsitoById(id) = esito  **context** Esito :: createEsito (id, nomeSudente, cognomeStudente, dataDiNascita, stringGP, validita)  **pre**: not findEsitoById(id)  **context** Esito :: updateEsito (e)  **pre**: findEsitoById(id) = esito  **context** Esito :: deleteEsito (id)  **pre**: findEsitoById(id) = esito |
| **Post-condizione** | **context** Esito :: findEsitoById(id)  **post**: findEsitoById(id) = esito  **context** Esito :: createEsito (id, nomeSudente, cognomeStudente, dataDiNascita, stringGP, validita)  **post**: findEsitoById(id) = esito  **context** Esito :: updateEsito (e)  **post**: findEsitoById(id) = esito  **context** Esito :: deleteEsito (id)  **post**: not findEsitoById(id) |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | **Dipartimento** |
| **Descrizione** | Questa classe modella oggetti Dipartimento. |
| **Pre-condizione** | **context**: Formato :: impostaFormato(formato)  **pre**: findFormato(id) = formato |
| **Post-condizione** | **context**: Formato :: impostaFormato(formato)  **post**: updateFormato(id) = formato |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | **DipartimentoDAO** |
| **Descrizione** | Questa classe contiene i metodi che permettono di effettuare operazioni CRUD per un Dipartimento. |
| **Pre-condizione** | **context** Dipartimento :: findDipartimentoById(codDip)  **pre**: findDipartimentoById(codDip) = dipartimento  **context** Dipartimento :: createDipartimento (nome)  **pre**: not findDipartimentoById(codDip)  **context** Dipartimento :: updateDipartimento(dip)  **pre**: findDipartimentoById(codDip) = dipartimento  **context** Dipartimento :: deleteDipartimento(dip)  **pre**: findDipartimentoById(codDip) = dipartimento |
| **Post-condizione** | **context** Dipartimento :: findDipartimentoById(codDip)  **post**: findDipartimentoById(codDip) = dipartimento  **context** Dipartimento :: createDipartimento (nome)  **post**: findDipartimentoById(codDip) = dipartimento  **context** Dipartimento :: updateDipartimento(dip)  **post**: findDipartimentoById(codDip) = dipartimento  **context** Dipartimento :: deleteDipartimento(dip)  **post**: not findDipartimentoById(codDip) |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | **Formato** |
| **Descrizione** | Questa classe è un semplice POJO che modella degli oggetti Formato. |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | **FormatoDAO** |
| **Descrizione** | Questa classe contiene i metodi che permettono di effettuare operazioni CRUD per un Formato. |
| **Pre-condizione** | **context** Formato :: findFormatoById(id)  **pre**: findFormatoById(id) = formato  **context** Formato :: createFormato (ddn, numGPvalidi, numGPnonValidi, numStudenti, nomeCognome)  **pre**: not findFormatoById(id)  **context** Formato :: updateDipartimento(formato)  **pre**: findFormatoById(id) = formato  **context** Formato :: deleteFormato(formato)  **pre**: findFormatoById(codDip) = formato |
| **Post-condizione** | **context** Formato :: findFormatoById(id)  **post**: findFormatoById(id) = formato  **context** Formato :: createFormato (ddn, numGPvalidi, numGPnonValidi, numStudenti, nomeCognome)  **post**: findFormatoById(id)= formato  **context** Formato :: updateDipartimento(formato)  **post**: findFormatoById(id) = formato  **context** Formato :: deleteFormato(formato)  **post**: not findFormatoById(codDip) = formato |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | **SessionController** |
| **Descrizione** | Questa classe si occupa di gestire le operazioni legate alla Sessione. |
| **Pre-condizione** | **context** void :: doGet (request, response)  **context** void doPost (request, response) |
| **Post-condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | **AccessController** |
| **Descrizione** | Questa classe contiene le operazioni necessarie alla gestione dei dati personali del Docente e del Direttore di Dipartimento. |
| **Pre-condizione** | **context** void :: doGet (request, response)  **context** void doPost (request, response) |
| **Post-condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | **ReportController** |
| **Descrizione** | Questa classe contiene le operazioni legate alla gestione dei Report. |
| **Pre-condizione** | **context** void :: doGet (request, response)  **context** void doPost (request, response) |
| **Post-condizione** |  |
| **Invarianti** |  |

1. Class Diagram
2. Glossario