**Template**

|  |  |
| --- | --- |
| **Outline** | **Description** |
| **1. Introduction** 1.1 Object design trade-offs 1.2 Interface documentation guidelines 1.3 Definitions, acronyms, and abbreviations 1.4 References | *The first section of the ODD is an introduction to the document. It describes the general trade-offs made by developers (e.g., buy vs. build, memory space vs. response time), guidelines and conventions (e.g., naming conventions, boundary cases, exception handling mechanisms), and an overview of the document. Interface documentation guidelines and coding conventions are the single most important factor that can improve communication between developers during object design. These include a list of rules that developers should use when designing and naming interfaces. These are examples of such conventions:*   * Classes are named with singular nouns. * Methods are named with verb phrases, fields, and parameters with noun phrases. * Error status is returned via an exception, not a return value. * Collections and containers have an elements() method returning an Enumeration. * *Enumerations returned by elements() methods are robust to element removals.*   Such conventions help developers design interfaces consistently, even if many developers contribute to the interface specification. Moreover, making these conventions explicit before object design makes it easier for developers to follow them. In general, these conventions should not evolve during the project. |
|  | |
| **2. Packages** | *The second section of the ODD, Packages, describes the decomposition of subsystems into packages and the file organization of the code. This includes an overview of each package, its dependencies with other packages, and its expected usage.* |
|  | |
| **3. Class interfaces Glossary** | *The third section, Class interfaces, describes the classes and their public interfaces. This includes an overview of each class, its dependencies with other classes and packages, its public attributes, operations, and the exceptions they can raise.* |

Progetto di Ingegneria del software

Anno accademico 2018/2019



*Carcheck*



|  |  |  |
| --- | --- | --- |
| **Tabella Componenti** | | |
| **Capriglione** | **Francesco** |  |
| **D’Auria** | **Aldo** |  |
| **De Falco** | **Daniele** |  |
| **Iacovazzo** | **Giovanni** | 051214774 |

**Indice**

**1. Introduzione**

1.1 Object design trade-off

1.2 Interface documentation guidelines

1.3 Definizioni, acronimi e abbreviazioni

1.4 Riferimenti

**2. Packages**

**3. Class Interface Glossary**

1.0 Introduzione

Nel seguente documento vengono definiti tutti gli aspetti analizzati prima di procedere all’implementazione. Nelle fasi precedenti sono stati definiti i vari requisiti su cui modellare il sistema CarCheck. Il nostro scopo qui è andare ad interfacciarci con i Solution Objects, e tenere conto di aspetti di aspetti significativi riguardanti tempo di esecuzione di un’operazione, utilizzo di memoria e altre misure di costo.

1.1 Object design trade-off

**Interfaccia vs Usabilità**

L’interfaccia, dovendo essere semplice da utilizzare ed intuitiva, utilizzerà una serie di componenti standard. Saranno inoltre presenti delle etichette, suggerimenti ed altri elementi che faranno capire all’utente che l’azione che sta eseguendo ha prodotto un risultato oppure è incorretta. Eventualmente potranno essere utilizzate delle legende per una maggiore compresione di simboli e notazioni.

**Spazio di memoria vs Tempo di risposta**

Si cercherà di mantenere un equilibrio tra questi due parametri. Infatti, da una parte abbiamo una grande quantità di informazioni da memorizzare su un numero molto alto di veicoli, dall’altra bisogna assicurare che i tempi delle risposte alle richieste degli utenti rispettino ciò che è stato espresso nella definizione dei requisiti non funzionali.

**Sicurezza vs efficienza**

Per gli utenti registrati sarà data priorità alla sicurezza. L’approccio utilizzato sarà quello di utilizzare come fattore di autenticazione una coppia Email-Password per evitare accessi non autorizzati. Altro aspetto, sempre relativo alla sicurezza, riguarda la criptazione dei dati sensibili trasmessi al server (email, password, alcuni dati relativi a revisioni ???????? ).

Per gli utenti non registrati tali problematiche non sono affatto presenti. La priorità è quindi data interamente all’efficienza.

**Prestazioni vs Costi:**

Essendo un progetto universitario e non avendo risorse finanziarie, tutte le tecnologie da noi scelte sono distribuite sotto licenza open-source o free-ware. Per questo motivo i costi non sono un aspetto da tenere in considerazione.

Per quanto riguarda invece le prestazioni, tali tecnologie forniscono comunque dei buoni risultati sotto diversi aspetti e per questo rappresentano una buona soluzione per la realizzazione del sistema.