

|  |  |  |  |
| --- | --- | --- | --- |
| [Composizione gruppo] |  |  |  |

SDD System Design Document

Dashboard Dipartimento Informatica

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Cambiamenti | Autori |
|  |  |  |  |
|  |  |  |  |

Sommario

1. [1. Introduzione 3](#_Toc436218379)

[1.1 Obiettivi del sistema](#_Toc436218380)

[1.2 Design Goals](#_Toc436218381)

[1.3 Definizioni, acronimi e abbreviazioni](#_Toc436218382)

[1.4 Riferimenti](#_Toc436218383)

[1.5 Panoramica](#_Toc436218384)

1. [2. Architettura del Sistema corrente 3](#_Toc436218385)
2. [3. Architettura del Sistema proposto 3](#_Toc436218386)

[3.1 Panoramica](#_Toc436218387)

[3.2 Decomposizione sottosistemi](#_Toc436218388)

[3.3 Mapping hardware/software](#_Toc436218389)

[3.4 Gestione dati persistenti](#_Toc436218390)

[3.5 Controllo degli accessi e sicurezza](#_Toc436218391)

[3.6 Controllo globale del software](#_Toc436218392)

[3.7 Condizione limite](#_Toc436218393)

1. [4. Servizi dei Sottosistemi 3](#_Toc436218394)
2. [Glossario 3](#_Toc436218395)

# 1. Introduzione

## 1.1 Obiettivi del sistema

## 1.2 Design Goals

Il Bruegge suggerisce diverse categorie di design goals:

* Criteri di performance
  + Tempo di risposta
  + Throughput
  + Memoria
* Criteri di affidabilità
  + Robustezza
  + Affidabilità
  + Disponibilità
  + Tolleranza ai guasti
  + Security
  + Safety
* Criteri di costo
  + Costi di sviluppo
  + Costi di amministrazione
  + Costi di distribuzione
  + Costi di mantenimento
  + Costi di aggiornamento
* Criteri di manutenzione
  + Estendibilità
  + Modificabilità
  + Adattabilità
  + Tracciabilità dei requisiti
  + Affidabilità
  + Portabilità
* Criteri utenti finali
  + Usabilità
  + Utilità

Spiegare come si intende raggiungere questi obiettivi di progettazione ELIMINANDO i design goal che non sono di interesse.

## 1.3 Definizioni, acronimi e abbreviazioni

## 1.4 Riferimenti

[Documenti, libri di testo o altro materiale usato per produrre questo documento]

## 1.5 Panoramica

[Breve descrizione della composizione del documento]

# 2. Architettura del Sistema corrente

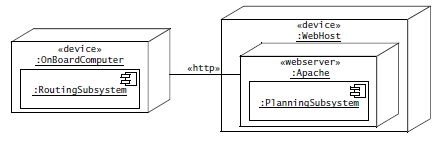
# 3. Architettura del Sistema proposto

## 3.1 Panoramica

## 3.2 Decomposizione in sottosistemi

[specificare l’architettura (es client- server). PRESTARE ATTENZIONE ALLA DIFFERENZA TRA LAYER E TIER. Ricordatevi IL DISEGNO che mostra le relazioni tra i sottosistemi.

Es diagramma di deploy in cui viene mostrato anche l’hardware su cui è stato fatto il deploy delle componenti software:

 ]

## 3.3 Mapping hardware/software

[specificare se verranno utilizzate componenti off- the- shelf (es DBMS MySQL)e i pattern che abbiamo intenzione di usare. ]

## 3.4 Gestione dati persistenti

[data la particolarità del nostro sistema le PM pensano sia utile elencare tutti gli indicatori con priorità associata. La priorità la indichiamo con tre colori diversi:

urgentissimi: da fare assolutamente

consigliato inserire: che sono da valutare se il tempo è sufficiente

trascurabili: utili, ma si possono evitare)

|  |  |  |
| --- | --- | --- |
| NOME INDICATORE | DATI | PRIORITÀ |
| [nome dell’indicatore secondo l’ANVUR] | [dati necessari] | [COLORE priorità] |
|  |  |  |

[PER GLI INDICATORI CHE ABBIAMO SCELTO DI IMPLEMENTARE elencarli e riassumere i dati in comune. Potrebbe essere utile nel nostro caso avere lo scema ER. ]

[spiegare la strategia di memorizzazione dei dati]

## 3.5 Controllo degli accessi e sicurezza

[elencare tutte le operazioni che l’utente può fare]

## 3.6 Controllo flusso globale del sistema

[Specificare se il controllo è

* Procedure- driven
* Guidato dagli eventi
* Guidato dai thread

]

## 3.7 Condizione limite

[CIT Bruegge:

*Boundary conditions:* How is the system initialized and shut down? How are

exceptional cases handled? System initialization and shutdown often represent much of

the complexity of a system, especially in a distributed environment. Initialization,

shutdown, and exception handling have an impact on the interface of *all subsystem*s.

**Configuration**. For each persistent object, we examine in which use cases it is created

or destroyed (or archived). For objects that are not created or destroyed in any of the

common use cases (e.g., Maps in the MyTrip system), we add a use case invoked by a

system administrator (e.g., ManageMaps in the MyTrip system).

**• Start-up and shutdown**. For each component (e.g., a WebServer), we add three use

cases to start, shutdown, and configure the component. Note that a single use case can

manage several tightly coupled components.

**• Exception handling**. For each type of component failure (e.g., network outage), we

decide how the system should react (e.g., inform users of the failure). We document

each of these decisions with an exceptional use case that extends the relevant common

uses cases identified during requirements elicitation. Note that, when tolerating the

effects of a failure, the handling of an exceptional condition can lead to changing the

system design instead of adding an exceptional use case. For example, the

RouteAssistant can completely download the Trip onto the car before the start of the

trip.]

# 4. Servizi dei Sottosistemi

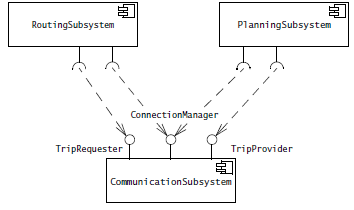


Diagramma delle component che esplicitano I servizi di ogni sottosistema e le interazioni tra gli stessi.

# Glossario

[Inserire qui tutti i termini utilizzati nel documento in modo da disambiguarli.]