**Technical Design**

**“Tu sì che mi capisci”**

**Table of content**

[**1**](#_heading=h.ihv636) **INTRODUCTION 3**

[**2**](#_heading=h.32hioqz) **PHYSICAL DATA DESIGN 4**

[*2.1*](#_heading=h.1hmsyys) *Data Dictionary 4*

[*2.2*](#_heading=h.41mghml) *Design of relational data bases (SQL) 4*

[2.2.1](#_heading=h.2grqrue) Table 1 (..n) – XXXXXXXXXXX 4

[2.2.1.1](#_heading=h.vx1227) Columns 4

[2.2.1.2](#_heading=h.3fwokq0) Keys 4

[2.2.1.3](#_heading=h.1v1yuxt) Event Actions 4

[2.2.1.4](#_heading=h.4f1mdlm) Stored procedures 5

[2.2.1.5](#_heading=h.2u6wntf) Sizes and other Physical properties 5

[2.2.1.6](#_heading=h.19c6y18) Indexes 5

[2.2.1.7](#_heading=h.3tbugp1) SQL Procedures for inserts and updates 5

[2.2.2](#_heading=h.28h4qwu) Relations 5

[*2.3*](#_heading=h.nmf14n) *File Design 5*

[2.3.1](#_heading=h.37m2jsg) File 1 (...n) – XXXXXXXXXXX 5

[2.3.1.1](#_heading=h.1mrcu09) File description 5

[2.3.1.2](#_heading=h.46r0co2) Fields 6

[**3**](#_heading=h.2lwamvv) **COMPONENT DESIGN 7**

[*3.1*](#_heading=h.111kx3o) *Type of components 7*

[3.1.1](#_heading=h.3l18frh) Details of software component types 7

[3.1.2](#_heading=h.206ipza) Rules and build recommendations 7

[*3.2*](#_heading=h.4k668n3) *Component inventory (structured design) 7*

[3.2.1](#_heading=h.2zbgiuw) Subsystem SSSSSSSSS (1...n) 7

[3.2.1.1](#_heading=h.1egqt2p) Function AAAAAAA (1...n) 7

[3.2.1.2](#_heading=h.2bn6wsx) Subsystem test cases 8

[3.2.2](#_heading=h.qsh70q) System test cases 8

[*3.3*](#_heading=h.3ygebqi) *Component Inventory (Object oriented design) 8*

[3.3.1](#_heading=h.2dlolyb) Subsystem or Layer SSSSSSS (1...n) 8

[3.3.1.1](#_heading=h.sqyw64) Class CCCCCCCCCC (1...n) 8

[3.3.1.2](#_heading=h.3cqmetx) Object XXXXXXXXXXX (1...n) 8

[3.3.1.3](#_heading=h.1rvwp1q) Component XXXXXXXXXXX (1...n) 8

[3.3.2](#_heading=h.3o7alnk) System test cases 8

1. **INTRODUCTION**

The Technical Design is based on the functional design. It extends the design specifications depending on the architecture and the real installation environment that are not part of the functional design.

The Technical Design document consists of 2 sections:

* Physical Data Design
* Component Design

Both parts are corresponding extensions of sections of the functional design:

* Logical Data Model
* Logical Process Model

*As well as in the functional design it is possible that any section of this template is not used in a special case. This depends on various factors. In this case IT SHOULD BE INFORMED “NOT APPLIED” o “(N/A)” in the body of the section or in the box if it is a table or a form, avoiding the deletion of the section, block or paragraph.*

*In case that the information was already included in another document, a reference or link to the section of this document should be included, AVOIDING TO REPEAT THE INFORMATION in any case.*

1. **PHYSICAL DATA DESIGN**

The physical data design links the physical specifications to the real architecture of the SQL tables and other files that are used to store permanently the information that is managed by the system. Not included is the storage of external data.

Nel nostro progetto salviamo le registrazioni della voce in base alle emozioni che servono per allenare l’intelligenza artificiale. L’applicazione si occuperà di salvare i dati relativi alle emozioni degli utenti per il fine statistico.

* 1. **Data Dictionary**

Design of the physical implementation of the data types, defined in the data dictionary that was prepared during the functional design phase, optimized depending on the system architecture and the interfaces. Contain the real data types, which are used in every functional data type.

|  |  |  |
| --- | --- | --- |
| DATA TYPE | DESCRIPTION | PHYSICAL CONFIGURATION |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. **Design of relational data bases (SQL)**

Contains information for the build of the tables like its columns (fields), primary keys, secondary keys, foreign keys, indexes, integrity rules, design of the stored procedures and the called events, relations between the tables, start up volume of the lines, yearly growth percentage, copy, restore and reorganisation procedures, etc.

Non siamo ancora a conoscenza del formato che andremo a utilizzare.

* + - 1. **Table 1 (..n) – Utente**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| Username | varchar | 20 |  | username dell’utente |
| Password | char | 32 |  | password utente |
| Email | varchar | 32 |  | mail utente |
| Immagine | varchar | 32 |  | url immagine utente |
| Xp | int | 8 |  | Xp accumulata dal utene |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | int autoincrement | 11 |

* + - 1. **Table 1 (..n) – VoltoRegistrato**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| Nome | varchar | 20 |  | nome del volto |
| Immagine | varchar | 32 |  | url immagine utente |
| IdUtente\* | integer | 11 |  | Utente che lo salva |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | int autoincrement | 11 |

* + - 1. **Table 1 (..n) – Dispositivo**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| Nome | varchar | 20 |  | nome del dispositivo |
| tipo | integer | 1 |  | flag per definire il tipo del dispositivo |
| Ip | varchar | 15 |  | ip dispositivi |
| Acceso | integer | 1 |  | flag per dire se il disp è attivo o no |
| IdUtente\* | integer | 11 |  | Utente che possiede il dispositivo |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | int autoincrement | 11 |

* + - 1. **Table 1 (..n) – VoltoTrovato**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| DataRilevazione | datatime |  |  | Data rilevazione |
| Vettore immagine | text |  |  | Vettore distanze tra le parti del volto |
| Immagine | varchar | 32 |  | url immagine utente |
| IdDispositivo\* | integer | 11 |  | Dispositivo che ha rilevato il volto |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | int autoincrement | 11 |

* + - 1. **Table 1 (..n) – EmozioneTrovata**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| IdEmozione\* | integer | 11 |  | Tipo di emozione rilevata |
| DataRilevazione | Date |  |  | Data in cui è stata rilevata l’emozione |
| IdDispositivo\* | integer | 11 |  | dispositivo che ha rilevato l’emozione |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | int autoincrement | 11 |

* + - 1. **Table 1 (..n) – Emozione**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| Tipo | integer | 2 |  | Tipo di emozione rilevata |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | int autoincrement | 11 |

* + - 1. **Table 1 (..n) – Quest**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| Concluso | integer | 1 |  | flag che identifica se è è stata completata |
| IdUtente\* | integer | 11 |  | chiave esterna che si riferisce all’utente |
| IdObiettivo\* | integer | 11 |  | chiave esterna che si riferisce al obiettivo dato |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | integer | 11 |

* + - 1. **Table 1 (..n) – Permesso**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| IdTrovato\* | integer | 11 |  | chiave primaria |
| IdVoltoRegistrato\* | integer | 11 |  | Volto riconoscuto |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| IdTrovato | int autoincrement | 11 |
| IdVoltoRegistrato | int autoincrement | 11 |

* + - 1. **Table 1 (..n) – Obiettivo**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| Immagine | varchar | 32 |  | url immagine obiettivo |
| Nome | varchar | 64 |  | Nome del obiettivo |
| Descrizione | varchar | 128 |  | Descrizione obiettivo |
| IdEmozione | Int | 11 |  | IdEmozione a cui fa riferimento |
| NRilevazioni | Int | 11 |  | Quante emozioni di quel tipo rilevare |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | integer autoincrement | 11 |

* + - 1. **Table 1 (..n) – Skill**

* + - * 1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| Id | integer | 11 |  | chiave primaria |
| Azione | varchar | 64 |  | Frase di attivazione |
| Descrizione | varchar | 128 |  | Descrizione skill |
| IdUtente\* | integer | 11 |  | Utente che la registra |
| IdEmozione\* | integer | 11 |  | Emozione che riguarda la skill |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| Id | int autoincrement | 11 |

* + - 1. **Table 1 (..n) – Reset**
         1. **Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD | DATA TYPE IN  DICTIONARY | LENGHT | VALIDATION RULES | COMMENTS |
| CodiceReset | integer | 11 |  | Codice da inserire per resettare la password |
| Email | varchar | 32 |  | IdUtente che ha fatto il recupero |
| Data | Datetime |  |  | Momento in cui ha fatto il reset |

* + - * 1. **Keys**

|  |  |  |
| --- | --- | --- |
| KEY | COMPOSITION | LENGTH |
| CodiceReset | int | 11 |

* + 1. **Relations**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RELATION ID. FROM FUNCTIONAL DESIGN | PRIMARY TABLE | KEY COLUMN IN PRIMARY TABLE | SECONDARY TABLE | KEY COLUMN IN SECONDARY TABLE | RESTRICTIONS AND INTEGRITY RULES |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. **COMPONENT DESIGN**

The component design contains the necessary information to plan and start the building of the system components.

The information is divided between information about the component type and the design of the system components organised as inventory and detail specifications.

* 1. **Type of components**

The information regarding to component types is meant to be a guide for the component construction. For this reason it includes or makes reference to any additional information about the construction, that wasn’t included or referenced in previous phases of requirements or design as guide or programming rules, tool documentation, etc.

* + 1. **Details of software component types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| COMPONENT TYPE | DETAILLED DESIGN | CONSTRUCTION TOOL | GENERATION OF EXECUTABLES | TEST WITH  COMPONENTS |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

* + 1. **Rules and build recommendations**

The following tables list the rules and building recommendation in line the clients standards and the other standards that were created by the Project team.

|  |  |  |
| --- | --- | --- |
| RULE OR  RECOMMENDATION | AFFECTED TYPE OF  COMPONENTS | COMMENTS |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. **Component inventory (structured design)**

The components of every subsystem and/or basic function are listed in detail.

* + 1. **Subsystem SSSSSSSSS (1...n)**
       1. **Function AAAAAAA (1...n)** 
          1. (Component XXXXXXXXXXX) (i...j)
* Unique name
* Component type
* Short Description
* Design(variable depending on the component type):
  + Diagram
  + Algorithm in pseudo-code
  + Input / Output
  + Verifications / error conditions
  + Calls to other modules
  + Access to tables and types: (Select, Insert, Delete, Update), columns for Where conditions in Selects, etc.
  + Access to files and types: (Sequential, Random), (Read, Write, Extend), etc,
* Component test cases
  + - * 1. Function test cases
      1. **Subsystem test cases**
    1. **System test cases**

* 1. **Component Inventory (Object oriented design)**

The components of every Subsystem are listed in detail.

* + 1. **Subsystem or Layer SSSSSSS (1...n)**
       1. **Class CCCCCCCCCC (1...n)**

…

* + - 1. **Object XXXXXXXXXXX (1...n)**

…

* + - 1. **Component XXXXXXXXXXX (1...n)**

…

* + 1. **System test cases**