PROGETTO DI PROGRAMMAZIONE AD OGGETTI
2015/2016

PCBuilder

Nome: Navid Taha
Matricola: 1069126

Ambiente di sviluppo

- Sistema operativo: Ubuntu 14.04
- Librerie Qt: 5.3.2
- Compilatore: GCC 4.6.1

Guida all'utilizzo

L'applicazione è in grado di memorizzare i dati dei database degli utenti e delle componenti tramite l'utilizzo di file xml, presenti all'interno della cartella in cui viene inserito l'eseguibile del progetto. I file vengono chiamati "Componenti.xml" e "Utenti.xml".
Vengono messi a disposizione entrambi i file .xml ed è possibile accedere utilizzando le credenziali seguenti (oppure, è possibile registrarsi normalmente):

Username	**Password**	**Tipologia**
Premium	123	UtentePremium
Normale	123	UtenteRegistrato

Scopo del progetto

Lo scopo del progetto è quello di fornire un'applicazione utile alla creazione di configurazioni di PC assemblati, tramite un'apposita interfaccia grafica messa a disposizione. Il sistema, quindi, offrirà un database contenente le componenti presenti sul mercato.

Ogni componente viene rappresentato da un nome univoco, prezzo, nome del produttore e una serie di altre specifiche tecniche individuate a seconda del tipo di componente. Le specifiche tecniche vengono rappresentate in formato di stringa, in modo da permettere una descrizione più specifica ed accurata per i singoli componenti.
Le componenti che potranno fare parte di una configurazione sono le seguenti:

- Memoria
- Scheda grafica
- Scheda madre
- Processore
- Archiviazione
- Alimentatore
- Dissipatore processore
- Case
- Unità ottica
- Monitor
- Sistema operativo
- Mouse
- Tastiera
- Cuffie
- Altoparlanti

Di conseguenza, ci sarà una gerarchia (non principale) rappresentata dalla classe base Componente, contenente i campi dato nome, prezzo e produttore e le classi derivate corrispondenti alle componenti sopra illustrate.

E’ presente una gerarchia di utenti, i quali potranno usufruire dell’applicativo a seconda della tipologia di account. Sono infatti presenti 3 tipologie di account: non registrato, registrato e premium.
Le funzionalità offerte a tali classi sono le seguenti:

- Utente non registrato:
  - Registrazione
  - Accesso
  - Creazione di una configurazione
- Utente registrato:
  - Creazione di una configurazione
  - Salvataggio della configurazione nel proprio account
  - Modifica del proprio profilo
  - Possibilità di aggiornare il proprio profilo a premium
  - Visualizzare le proprie configurazione e modificarle

    - Eliminare il proprio account
    - Logout
- Utente premium:
    - Funzionalità offerte all'utente registrato
    - Possibilità di consultare la lista delle componenti presenti nel database e di visualizzarne le specifiche

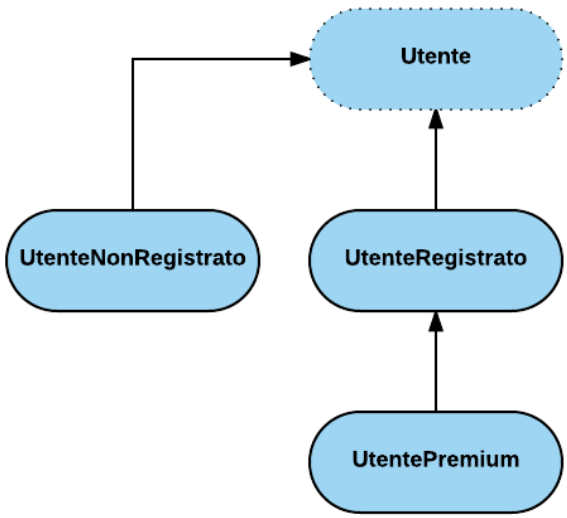
E' presente anche una sezione dedicata all'amministrazione dell'applicazione, con le seguenti funzionalità:

- Gestione componenti:
    - Aggiunta di un nuovo componente
    - Rimozione di un componente
- Gestione utenti:
    - Modifica dati di un utente
    - Eliminazione di un utente

Gerarchia delle classi

Gerarchia degli utenti

La gerarchia principale del progetto, viene rappresentata dalla classe base Utente e dalle classi derivate UtenteNonRegistrato, UtenteRegistrato e UtentePremium.



Utente:

La classe Utente rappresenta la classe base astratta ed è caratterizzata dai seguenti campi dato e metodi:

- Campi dato:
  - PCBuilderController* controller : puntatore al controller
- Metodi:
  - Utente(PCBuilderController*) : costruttore ad un parametro, che assume in input il puntatore al controller;
  - PCBuilderController* get_controller() const : metodo utile a ricavare il puntatore controller;
  - virtual void *showIndex*() const =0 : metodo virtuale puro utile a mostrare il corretto widget rappresentante l'index dell'utente a seconda del tipo di utente;
  - virtual Login* *get_Login*() const =0 : metodo virtuale puro utile a ricavare il puntatore alla classe rappresentante il Login, nel caso in cui sia presente;
  - virtual Profilo* *get_Profilo*() const =0 : metodo virtuale puro utile a ricavare il puntatore alla classe rappresentante il Profilo, nel caso in cui sia presente;
  - virtual ~*Utente*(){} : distruttore virtuale;
  - virtual QString *get_type*() const =0 : metodo virtuale pure utile a ricavare una stringa rappresentate il tipo dell'utente;
  - virtual QVector<Configurazione*>* *getConfigurazioni*() const =0 : metodo virtuale puro utile a ricavare il puntatore al vector contenente le configurazioni, in caso sia presente;
  - virtual Utente* clone() const =0 : metodo virtuale puro utile ad eseguire una copia di un oggetto puntato da un puntatore polimorfo.

UtenteNonRegistrato:

La classe UtenteNonRegistrato rappresenta una classe concreta derivata dalla classe Utente, ed è caratterizzata dai seguenti campi dato e metodi:

- Metodi:
  - UtenteNonRegistrato(PCBuilderController*) : costruttore ad un parametro, che assume in input il puntatore al controller;
  - virtual void *showIndex*() const : overriding del metodo virtuale puro della classe base;
  - virtual ~*UtenteNonRegistrato*(){}: distruttore virtuale;

    - virtual Login* *get_Login*() const : overriding del metodo virtuale puro della classe base;
    - virtual Profilo* *get_Profilo*() const : overriding del metodo virtuale puro della classe base;
    - virtual QVector<Configurazione*> *getConfigurazioni*() const :
    - virtual QString *get_type*() const : overriding del metodo virtuale puro della classe base;
    - virtual Utente* clone() const : overriding del metodo virtuale puro della classe base.

UtenteRegistrato:

La classe UtenteRegistrato rappresenta una classe concreta derivata dalla classe Utente, ed è caratterizzata dai seguenti campi dato e metodi:

- Campi dato:
    - Login id : classe contenente le credenziali di accesso;
    - Profilo pf : classe contenente le informazioni del profilo;
    - QVector<Configurazione*>* configurazioni : puntatore ad un vector contenente le configurazioni.
- Metodi:
    - UtenteRegistrato(PCBuilderController*, const Login&, const Profilo&) : costruttore a tre parametri, che assume in input il puntatore al controller, un oggetto Login contenente le credenziali di accesso e un oggetto Profilo contenente le informazioni del profilo;
    - virtual void *showIndex*() const : overriding del metodo virtuale puro della classe base;
    - virtual Login* *get_Profilo*() const : overriding del metodo virtuale puro della classe base;
    - virtual Profilo* *get_Profilo*() const : overriding del metodo virtuale puro della classe base;
    - virtual *~UtenteRegistrato*(){} : distruttore virtuale;
    - virtual QString *get_type*() const : overriding del metodo virtuale puro della classe base;
    - virtual QVector<Configurazione*>* *getConfigurazione*() const : overriding del metodo virtuale puro della classe base;
    - virtual Utente* clone() const : overriding del metodo virtuale puro della classe base.

UtentePremium:

La classe UtentePremium rappresenta una classe concreta derivata dalla classe UtenteRegistrato, ed è caratterizzata dai seguenti campi dato e metodi:

- Metodi:
  - UtentePremium(PCBuilderController*, const Login&, const Profilo&) : costruttore a tre parametri, che assume in input il puntatore al controller, un oggetto Login contenente le credenziali di accesso e un oggetto Profilo contenente le informazioni del profilo;
  - virtual ~*UtentePremium*(){} : distruttore virtuale;
  - virtual void *showIndex*() const : overriding del metodo virtuale puro della classe base;
  - virtual QString *get_type*() const : overriding del metodo virtuale puro della classe base;
  - virtual Utente* clone() const : overriding del metodo virtuale puro della classe base.

GUI

L'applicazione è stata sviluppata cercando di aderire il più possibile al design pattern MVC (Model View Controller), quindi separando il codice relativo alla business logic dal codice relativo all'interfaccia grafica, mediante l'utilizzo di un controller che funge da tramite.
L'interfaccia grafica si serve di una componente MainWindow, la quale rappresenta la finestra dell'applicazione e che fornisce come widget centrale un QStackedWidget. Quest'ultimo permette di spostarsi da una pagina all'altra in modo semplice. Inoltre, nella MainWindow sono presenti gli slot utili a cambiare pagina.
Le diverse pagine sono create mediante le classi che iniziano con "MyWidget" e nello specifico sono le seguenti:

- **MyWidget_IndexPCBuilder**: rappresenta la pagina iniziale dell'applicazione e fornisce i seguenti bottoni
  - Utente: permette di aprire l'index dell'utente e più in generale di accedere all'ambito utente;
  - Admin: permette di aprire l'index dell'amministratore e più in generale di accedere all'ambito amministratore.
- **MyWidget_IndexAdminPCBuilder**: rappresenta la pagina index dell'amministratore e fornisce i seguenti bottoni:
  - Gestisci componenti: permette di accedere alla gestione dei componenti;
  - Gestisci utenti: permette di accedere alla gestione degli utenti.

- **MyWidget_GestisciComponentiPCBuilder**: rappresenta la pagina da cui è possibile aggiungere o rimuovere un componente e fornisce i seguenti bottoni:
  - Aggiungi: permette di aggiungere un componente;
  - Rimuovi: permette di rimuovere un componente.
- **MyWidget_GestisciUtentiPCBuilder**: rappresenta la pagina da cui è possibile modificare ed eliminare un utente e fornisce i seguenti bottoni:
  - Modifica: permette di modificare i dati di un utente;
  - Elimina utente: permette di eliminare un utente.
- **MyWidget_IndexUtenteNonRegistratoPCBuilder**: rappresenta la pagina index dell'utente non registrato al sistema e fornisce i seguenti bottoni:
  - Crea una configurazione: permette di accedere alla pagina di creazione di una configurazione di PC assemblato;
  - Accedi: permette di eseguire l'accesso al sistema;
  - Registrati: permette di eseguire la registrazione al sistema.
- **MyWidget_RegistrazioneUtentePCBuilder**: rappresenta la pagina da cui è possibile effettuare la registrazione al sistema e fornisce i seguenti bottoni:
  - Registrati: permette di eseguire la registrazione, nel caso in cui i dati inseriti siano validi.
- **MyWidget_CreaConfPCBuilder**: rappresenta la pagina da cui è possibile creare una configurazione di PC assemblato e fornisce i seguenti bottoni:
  - Aggiungi: viene fornito per ogni tipo di componente che può formare una configurazione e permette di accedere alla lista dei componenti di tale tipologia e di aggiungerne uno;
  - Nuova configurazione: permette di resettare la configurazione, in modo da iniziarne una nuova;
  - Salva la configurazione: nel caso in cui si è acceduto al sistema, permette di salvare la propria configurazione nel sistema, altrimenti viene richiesto di effettuare l'accesso.
- **MyWidget_LoginPopUpPCBuilder**: rappresenta la finestra pop up che viene mostrata nel caso in cui si cerca di salvare la configurazione quando non si è eseguito l'accesso, fornisce i seguenti bottoni:
  - Accedi: permette di eseguire l'accesso al sistema.
- **MyWidget_PartsListPCBuilder**: rappresenta la pagina in cui viene mostrata la lista dei componenti di una certa tipologia e fornisce i seguenti bottoni:
  - Aggiungi: permette di aggiungere il componente specifico alla configurazione.
- **MyWidget_IndexUtenteRegistratoPCBuilder**: rappresenta la pagina index dell'utente registrato e fornisce i seguenti bottoni:
  - Crea una configurazione: permette di accedere alla pagina di creazione di una configurazione di PC assemblato;
  - Esci: permette di eseguire il logout;
  - Mostra configurazione salvate: permette di visualizzare le proprie configurazione salvate;

    - Modifica profilo: permette di modificare il proprio profilo.
- **MyWidget_ModificaProfiloUtentePCBuilder**: rappresenta la pagina da cui è possibile eliminare il proprio account, aggiornare lo stato del proprio account a premium, oppure modificare i dati del proprio account e fornisce i seguenti bottoni:
    - Elimina account: permette di eliminare definitivamente il proprio account dal sistema;
    - Diventa premium: permette di eseguire il passaggio alla versione premium dell'account per un tempo limitato;
    - Modifica: permette di modificare i dati del proprio account.
- **MyWidget_LeMieConfigurazioniPCBuilder**: rappresenta la pagina in cui vengono visualizzate le proprie configurazioni salvate precedentemente e fornisce i seguenti bottoni:
    - Modifica: permette di caricare la pagina di creazione di una configurazione con i dati prestabiliti dalla configurazione che si vuole modificare e quando la si salva, nel caso in cui si inserisce lo stesso nome precedente, essa verrà sostituita da quella modificata.
- **MyWidget_ConfigurazionePCBuilder**: rappresenta la pagina in cui viene visualizzata una configurazione.
- **MyWidget_IndexUtentePremiumPCBuilder**: rappresenta la pagina index dell'utente premium e fornisce i seguenti bottoni:
    - Crea una configurazione: permette di accedere alla pagina di creazione di una configurazione di PC assemblato;
    - Esci: permette di eseguire il logout;
    - Mostra configurazione salvate: permette di visualizzare le proprie configurazione salvate;
    - Modifica profilo: permette di modificare il proprio profilo;
    - Cerca tra le parti: permette di accedere alla pagina da cui è possibile cercare i componenti singoli e consultarne le specifiche.
- **MyWidget_CercaPartiPCBuilder**: rappresenta la pagina che consente di visualizzare una lista di tutte le componenti di un dato tipo e fornisce i seguenti bottoni:
    - Specifiche: viene fornito per ogni componente e permette di visualizzarne le specifiche tecniche.
- **MyWidget_SpecificheComponentePCBuilder**: rappresenta la pagina che consente di visualizzare le specifiche tecniche di un dato componente.