

EasyTicket

Generated by Doxygen 1.9.1

1 Documentation EasyTicket	1
1.1 Version	1
1.1.1 v1.1	1
1.1.2 v1.2	1
1.1.3 v1.3	1
1.1.4 v1.4	1
1.1.5 v1.5	1
1.1.6 v1.6	2
1.1.7 v1.7	2
1.1.8 v1.8	2
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 Namespace Documentation	9
5.1 Ui Namespace Reference	9
5.1.1 Detailed Description	9
6 Class Documentation	11
6.1 Admin Class Reference	11
6.1.1 Detailed Description	11
6.1.2 Constructor & Destructor Documentation	12
6.1.2.1 Admin()	12
6.1.3 Member Function Documentation	12
6.1.3.1 estUnAdmin()	12
6.2 Categorie Class Reference	12
6.2.1 Detailed Description	13
6.2.2 Constructor & Destructor Documentation	13
6.2.2.1 Categorie()	13
6.2.3 Member Function Documentation	13
6.2.3.1 getID()	14
6.2.3.2 getNom()	14
6.3 Client Class Reference	14
6.3.1 Detailed Description	15
6.3.2 Constructor & Destructor Documentation	15
6.3.2.1 Client()	15
6.3.3 Member Function Documentation	15
6.3.3.1 estUnClient()	15

6.4 GestionnaireBDD Class Reference	16
6.4.1 Detailed Description	16
6.4.2 Member Function Documentation	17
6.4.2.1 authentifier()	17
6.4.2.2 enregistrer_ticket()	17
6.4.2.3 select()	17
6.4.2.4 setComboBox()	18
6.4.2.5 type_utilisateur()	18
6.5 GestionnaireDialogue Class Reference	18
6.5.1 Detailed Description	19
6.5.2 Member Function Documentation	19
6.5.2.1 authentication()	19
6.5.2.2 enregistrer_ticket()	20
6.5.2.3 setComboBox()	20
6.5.2.4 typeUtilisateur()	20
6.5.2.5 verificationMessage()	21
6.6 Ingenieur Class Reference	21
6.6.1 Detailed Description	22
6.6.2 Constructor & Destructor Documentation	22
6.6.2.1 Ingenieur()	22
6.6.3 Member Function Documentation	22
6.6.3.1 estUnIngenieur()	22
6.7 Logiciel Class Reference	23
6.7.1 Detailed Description	23
6.7.2 Constructor & Destructor Documentation	23
6.7.2.1 Logiciel()	23
6.7.3 Member Function Documentation	24
6.7.3.1 getID()	24
6.7.3.2 getNom()	24
6.8 MainWindow Class Reference	24
6.8.1 Constructor & Destructor Documentation	25
6.8.1.1 MainWindow()	25
6.8.2 Member Function Documentation	25
6.8.2.1 setGestionnaireDialogue()	25
6.9 Message Class Reference	26
6.9.1 Detailed Description	27
6.9.2 Constructor & Destructor Documentation	27
6.9.2.1 Message()	27
6.9.3 Member Function Documentation	27
6.9.3.1 getDate_envoi()	27
6.9.3.2 getId_message()	28
6.9.3.3 getMessage()	28

6.9.3.4	getTicket()	28
6.9.3.5	getUser()	28
6.9.3.6	setDate_envoi()	28
6.9.3.7	setId_message()	29
6.9.3.8	setMessage()	29
6.9.3.9	setTicket()	29
6.9.3.10	setUser()	30
6.9.3.11	toString()	30
6.10	PageAccueilClient Class Reference	30
6.10.1	Constructor & Destructor Documentation	31
6.10.1.1	PageAccueilClient()	31
6.10.2	Member Function Documentation	31
6.10.2.1	setClient()	31
6.11	PageAjoutTicket Class Reference	31
6.11.1	Constructor & Destructor Documentation	32
6.11.1.1	PageAjoutTicket()	32
6.11.2	Member Function Documentation	32
6.11.2.1	setClient()	32
6.12	PageLogin Class Reference	33
6.12.1	Constructor & Destructor Documentation	33
6.12.1.1	PageLogin()	33
6.13	Personnel Class Reference	34
6.13.1	Detailed Description	34
6.13.2	Constructor & Destructor Documentation	34
6.13.2.1	Personnel()	35
6.13.3	Member Function Documentation	35
6.13.3.1	estUnPersonnel()	35
6.14	Système Class Reference	35
6.14.1	Detailed Description	36
6.14.2	Constructor & Destructor Documentation	36
6.14.2.1	Système()	36
6.14.3	Member Function Documentation	37
6.14.3.1	getId_système()	37
6.14.3.2	getNom()	37
6.14.3.3	setId_système()	37
6.14.3.4	setNom()	37
6.15	Technicien Class Reference	38
6.15.1	Detailed Description	38
6.15.2	Constructor & Destructor Documentation	39
6.15.2.1	Technicien()	39
6.15.3	Member Function Documentation	39
6.15.3.1	estUnTechnicien()	39

6.16 Ticket Class Reference	40
6.16.1 Detailed Description	41
6.16.2 Constructor & Destructor Documentation	41
6.16.2.1 Ticket()	41
6.16.3 Member Function Documentation	41
6.16.3.1 addMessage()	41
6.16.3.2 getClient()	42
6.16.3.3 getDate_creation()	42
6.16.3.4 getDate_fermeture()	42
6.16.3.5 getId_ticket()	42
6.16.3.6 getListeMessages()	43
6.16.3.7 getLogiciel()	43
6.16.3.8 getPersonnel()	43
6.16.3.9 getSysteme()	43
6.16.3.10 setClient()	43
6.16.3.11 setDate_creation()	44
6.16.3.12 setDate_fermeture()	44
6.16.3.13 setId_ticket()	44
6.16.3.14 setListeMessages()	45
6.16.3.15 setLogiciel()	45
6.16.3.16 setPersonnel()	45
6.16.3.17 setSysteme()	45
6.17 Utilisateur Class Reference	46
6.17.1 Detailed Description	47
6.17.2 Constructor & Destructor Documentation	48
6.17.2.1 Utilisateur()	48
6.17.3 Member Function Documentation	48
6.17.3.1 estUnAdmin()	48
6.17.3.2 estUnClient()	48
6.17.3.3 estUnIngenieur()	49
6.17.3.4 estUnPersonnel()	49
6.17.3.5 estUnTechnicien()	49
6.17.3.6 getID()	49
6.17.3.7 getMail()	50
6.17.3.8 getMDP()	50
6.17.3.9 getNom()	50
6.17.3.10 getPrenom()	50
6.17.3.11 setID()	50
6.17.3.12 setMail()	51
6.17.3.13 setMDP()	51
6.17.3.14 setNom()	51
6.17.3.15 setPrenom()	52

Chapter 1

Documentation EasyTicket

1.1 Version

1.1.1 v1.1

Page de connexion avec au moment de la connexion, la création d'un objet [Utilisateur](#).

Peut importe le login et le mot de passe, on affiche un message contenant le 'toString' de notre objet [Utilisateur](#).

1.1.2 v1.2

Une fois connecté avec un compte 'client', l'utilisateur accède à la page d'accueil ([PageAccueilClient](#)).

Ajout d'un [GestionnaireDialogue](#) qui fait le lien entre l'utilisateur et les données.

1.1.3 v1.3

Le bouton de création de ticket mène à une page vierge qui servira pour créer les tickets ([PageAjoutTicket](#)).

Ajout d'un bouton de retour à la page d'accueil quand on est sur la page de création de ticket.

1.1.4 v1.4

Création de la mise en page du formulaire pour créer le ticket.

Affiche le resultat sur la sortie standard et si jamais il y a des champs invalides.

1.1.5 v1.5

Vérification des champs saisis par l'utilisation lors de la création d'un nouveau ticket.

Fonctions de vérifications pour le [Message](#) du ticket et génération d'un message contenant le descriptif du [Ticket](#) créé (sans encore le message).

1.1.6 v1.6

Implémentation de l'ajout de [Message](#) dans le [Ticket](#) au moment de sa création. Au moment du clic pour créer le [Ticket](#), on affiche les données du [Ticket](#) ainsi que le [Message](#) ajouté.

Si tout est ok, on retourne à la page d'accueil ([PageAccueilClient](#)).

1.1.7 v1.7

Ajout de la base de donnée au format db avec SQLite.

Ajout d'une classe [GestionnaireBDD](#) pour gérer l'interfaçage avec le bdd.

- Importer la bdd.
- Faire un select sur la bdd, etc.

1.1.8 v1.8

Ajout des fonctions pour communiquer avec la bdd :

- Au niveau de la connexion.
- et de l'enregistrement d'un ticket et du message.

Création des ComboBox en fonction des tables de la BDD.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Ui	Classe correspondant à la fenêtre principale de l'application EasyTicket	9
--------------------	--	-------------------

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Categorie	12
GestionnaireBDD	16
GestionnaireDialogue	18
Logiciel	23
Message	26
QMainWindow	
MainWindow	24
QWidget	
PageAccueilClient	30
PageAjoutTicket	31
PageLogin	33
Systeme	35
Ticket	40
Utilisateur	46
Admin	11
Client	14
Personnel	34
Ingenieur	21
Technicien	38

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Admin	La classe Admin permet de décrire un utilisateur avec les droits administrateurs sur le programme EasyTicket	11
Categorie	La classe Categorie permet de décrire les différentes catégories d'un Ticket	12
Client	La classe Client permet de décrire un utilisateur avec les droits clients sur le programme Easy↔Ticket	14
GestionnaireBDD	La classe GestionnaireBDD correspond à l'interface de communication entre les données du projet et le GestionnaireDialogue	16
GestionnaireDialogue	La classe GestionnaireDialogue correspond à l'interface de communication entre l' Utilisateur et le modèle	18
Ingenieur	La classe Ingenieur permet de décrire un utilisateur appartenant au Personnel avec les droits ingénieur sur le programme EasyTicket	21
Logiciel	La classe Logiciel permet de décrire un logiciel dans l'application EasyTicket	23
MainWindow	24
Message	La classe Message est la classe correspondant à un Message sur un Ticket	26
PageAccueilClient	30
PageAjoutTicket	31
PageLogin	33
Personnel	La classe Personnel est une classe abstraite qui permet de décrire l'ensemble des Utilisateur appartenant au personnel dans l'application EasyTicket	34
Systeme	La classe Systeme permet de décrire un système d'exploitation dans l'application EasyTicket	35
Technicien	La classe Technicien permet de décrire un utilisateur appartenant au Personnel avec les droits technicien sur le programme EasyTicket	38
Ticket	La classe Ticket permet de décrire un ticket dans l'application EasyTicket	40
Utilisateur	La classe Utilisateur est une classe abstraite qui permet de décrire l'ensemble des utilisateurs dans l'application EasyTicket	46

Chapter 5

Namespace Documentation

5.1 Ui Namespace Reference

Classe correspondant à la fenêtre principale de l'application EasyTicket.

5.1.1 Detailed Description

Classe correspondant à la fenêtre principale de l'application EasyTicket.

La classe [PageLogin](#) correspond à la vue de la page de connexion pour les [Utilisateur](#).

La classe [PageAjoutTicket](#) correspond à la vue de la page qui permet au [Client](#) connecté de créer un nouveau [Ticket](#).

La classe [PageAccueilClient](#) correspond à la vue de la page d'accueil pour les [Client](#).

Authors

Nicolas Robert, Victor Dallé, Claire Kurth

Version

1.3

Author

Victor Dallé, Nicolas Robert, Claire Kurth

Version

1.5

Author

Victor Dallé, Nicolas Robert, Claire Kurth

Version

1.2

Chapter 6

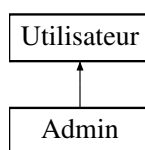
Class Documentation

6.1 Admin Class Reference

La classe [Admin](#) permet de décrire un utilisateur avec les droits administrateurs sur le programme EasyTicket.

```
#include <admin.h>
```

Inheritance diagram for Admin:



Public Member Functions

- [Admin](#) ()
Constructeur par défaut de la classe.
- [Admin](#) (QString [id](#), QString [nom](#), QString [prenom](#), QString [mdp](#), QString [mail](#))
Constructeur utilisant les données personnelles de l'admin.
- [~Admin](#) ()
Destructeur de la classe.
- bool [estUnAdmin](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Admin](#).

Additional Inherited Members

6.1.1 Detailed Description

La classe [Admin](#) permet de décrire un utilisateur avec les droits administrateurs sur le programme EasyTicket.

Authors

Claire Kurth, Nicolas Robert, Victor Dallé

Version

1.1

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Admin()

```
Admin::Admin (
    QString id,
    QString nom,
    QString prenom,
    QString mdp,
    QString mail )
```

Constructeur utilisant les données personnelles de l'admin.

Parameters

<i>id</i>	L'identifiant de l'admin.
<i>nom</i>	Le nom de l'admin.
<i>prenom</i>	Le prénom de l'admin.
<i>mdp</i>	Le mot de passe de l'admin.
<i>mail</i>	Le mail de l'admin.

6.1.3 Member Function Documentation

6.1.3.1 estUnAdmin()

```
bool Admin::estUnAdmin ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Admin](#).

Returns

true

Reimplemented from [Utilisateur](#).

The documentation for this class was generated from the following files:

- include/admin.h
- src/admin.cpp

6.2 Categorie Class Reference

La classe [Categorie](#) permet de décrire les différentes catégories d'un [Ticket](#).

```
#include <categorie.h>
```

Public Member Functions

- [Categorie](#) ()
Constructeur par défaut.
- [Categorie](#) (int id_categorie, QString nom_categorie)
Constructeur de la classe Catégorie.
- QString [getNom](#) ()
Méthode permettant de récupérer le nom de la [Categorie](#).
- int [getID](#) ()
Méthode retournant l'id de la catégorie.

6.2.1 Detailed Description

La classe [Categorie](#) permet de décrire les différentes catégories d'un [Ticket](#).

Authors

Nicolas Robert, Victor Dallé, Claire Kurth

Version

1.5

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Categorie()

```
Categorie::Categorie (
    int id_categorie,
    QString nom_categorie )
```

Constructeur de la classe Catégorie.

Parameters

<i>id_categorie</i>	l'identifiant de la catégorie
<i>nom_categorie</i>	le nom de la catégorie

6.2.3 Member Function Documentation

6.2.3.1 getID()

```
int Categorie::getID ( )
```

Méthode retournant l'id de la catégorie.

Returns

L'id de la categorie.

6.2.3.2 getNom()

```
QString Categorie::getNom ( )
```

Méthode permettant de récupérer le nom de la [Categorie](#).

Returns

nom_categorie

The documentation for this class was generated from the following files:

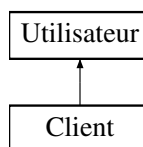
- include/categorie.h
- src/categorie.cpp

6.3 Client Class Reference

La classe [Client](#) permet de décrire un utilisateur avec les droits clients sur le programme EasyTicket.

```
#include <client.h>
```

Inheritance diagram for Client:



Public Member Functions

- [Client](#) ()
Constructeur par défaut de la classe.
- [Client](#) (QString [id](#), QString [nom](#), QString [prenom](#), QString [mdp](#), QString [mail](#))
Constructeur utilisant les données personnelles du client.
- [~Client](#) ()
Destructeur de la classe.
- bool [estUnClient](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Client](#).

Additional Inherited Members

6.3.1 Detailed Description

La classe [Client](#) permet de décrire un utilisateur avec les droits clients sur le programme EasyTicket.

Authors

Claire Kurth, Nicolas Robert, Victor Dallé

Version

1.1

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Client()

```
Client::Client (
    QString id,
    QString nom,
    QString prenom,
    QString mdp,
    QString mail )
```

Constructeur utilisant les données personnelles du client.

Parameters

<i>id</i>	L'identifiant du client.
<i>nom</i>	Le nom du client.
<i>prenom</i>	Le prénom du client.
<i>mdp</i>	Le mot de passe du client.
<i>mail</i>	Le mail du client.

6.3.3 Member Function Documentation

6.3.3.1 estUnClient()

```
bool Client::estUnClient ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Client](#).

Returns

true

Reimplemented from [Utilisateur](#).

The documentation for this class was generated from the following files:

- include/client.h
- src/client.cpp

6.4 GestionnaireBDD Class Reference

La classe [GestionnaireBDD](#) correspond à l'interface de communication entre les données du projet et le [GestionnaireDialogue](#).

```
#include <gestionnairebdd.h>
```

Public Member Functions

- [GestionnaireBDD](#) ()
Constructeur par défaut.
- void [select](#) (QString query)
Méthode qui permet d'afficher le résultat d'une requete passée en paramètre.
- [Client authentifier](#) (QString id, QString mdp)
Méthode qui permet d'authentifier un client.
- QString [type_utilisateur](#) (QString id_utilisateur, QString mdp)
Méthode qui permet de récupérer la catégorie d'un [Utilisateur](#).
- void [enregistrer_ticket](#) ([Ticket](#) &ticket)
Méthode qui permet d'enregistrer un ticket dans la BDD.
- void [setComboBox](#) (QComboBox *box, QString type)
Méthode qui remplit les comboBox de la page [PageAjoutTicket](#) en fct de la BDD.

Protected Attributes

- QSqlDatabase [my_db](#)
La base de donnée SQLite.

6.4.1 Detailed Description

La classe [GestionnaireBDD](#) correspond à l'interface de communication entre les données du projet et le [GestionnaireDialogue](#).

Authors

Nicolas Robert, Victor Dallé, Claire Kurth

Version

1.7

6.4.2 Member Function Documentation

6.4.2.1 authentifier()

```
Client GestionnaireBDD::authentifier (
    QString id,
    QString mdp )
```

Méthode qui permet d'authentifier un client.

Parameters

<i>id</i>	L'identifiant du client.
<i>mdp</i>	Le mot de passe du client.

Returns

le [Client](#) identifié.

6.4.2.2 enregistrer_ticket()

```
void GestionnaireBDD::enregistrer_ticket (
    Ticket & ticket )
```

Méthode qui permet d'enregistrer un ticket dans la BDD.

Parameters

<i>ticket</i>	Le ticket à enregistrer.
---------------	--------------------------

6.4.2.3 select()

```
void GestionnaireBDD::select (
    QString query )
```

Méthode qui permet d'afficher le résultat d'une requete passée en paramètre.

Parameters

<i>query</i>	La requête à envoyer à la bdd.
--------------	--------------------------------

6.4.2.4 setComboBox()

```
void GestionnaireBDD::setComboBox (
    QComboBox * box,
    QString type )
```

Méthode qui remplit les comboBox de la page [PageAjoutTicket](#) en fct de la BDD.

Parameters

<i>box</i>	La ComboBox.
<i>type</i>	Le type de donnée de la ComboBox (categorie, systeme, logiciel).

6.4.2.5 type_utilisateur()

```
QString GestionnaireBDD::type_utilisateur (
    QString id_utilisateur,
    QString mdp )
```

Méthode qui permet de récupérer la catégorie d'un [Utilisateur](#).

Parameters

<i>id_utilisateur</i>	l'identifiant de l' Utilisateur .
<i>mdp</i>	le mot de passe de l' Utilisateur .

Returns

la catégorie de l'utilisateur s'il existe, null sinon

The documentation for this class was generated from the following files:

- include/gestionnairebdd.h
- src/gestionnairebdd.cpp

6.5 GestionnaireDialogue Class Reference

La classe [GestionnaireDialogue](#) correspond à l'interface de communication entre l'[Utilisateur](#) et le modèle.

```
#include <gestionnairedialogue.h>
```

Public Member Functions

- [GestionnaireDialogue](#) ()
Constructeur par défaut.
- QString [typeUtilisateur](#) (QString id, QString mdp)
Méthode qui permet d'identifier le type d'[Utilisateur](#) qui se connecte si il existe.
- [Client authentication](#) (QString id, QString mdp)
Méthode qui permet d'identifier un [Client](#).
- bool [verificationMessage](#) (QString message)
Méthode permettant de vérifier que le message est bon.
- void [setComboBox](#) (QComboBox *box, QString type)
Méthode qui remplit les comboBox de la page [PageAjoutTicket](#) en fct de la BDD.
- void [enregistrer_ticket](#) ([Ticket](#) ticket)
Méthode qui permet d'enregistrer le ticket dans la BDD.

6.5.1 Detailed Description

La classe [GestionnaireDialogue](#) correspond à l'interface de communication entre l'[Utilisateur](#) et le modèle.

Authors

Nicolas Robert, Victor Dallé, Claire Kurth

Version

1.5

6.5.2 Member Function Documentation

6.5.2.1 authentication()

```
Client GestionnaireDialogue::authentication (
    QString id,
    QString mdp )
```

Méthode qui permet d'identifier un [Client](#).

Parameters

<i>id</i>	L'identifiant du Client .
<i>mdp</i>	Le mot de passe du Client .

Returns

le [Client](#) identifié.

6.5.2.2 enregistrer_ticket()

```
void GestionnaireDialogue::enregistrer_ticket (
    Ticket ticket )
```

Méthode qui permet d'enregistrer le ticket dans la BDD.

Parameters

<i>ticket</i>	Le ticket à enregistrer.
---------------	--------------------------

6.5.2.3 setComboBox()

```
void GestionnaireDialogue::setComboBox (
    QComboBox * box,
    QString type )
```

Méthode qui remplit les comboBox de la page [PageAjoutTicket](#) en fct de la BDD.

Parameters

<i>box</i>	La ComboBox.
<i>type</i>	Le type de donnée de la ComboBox (categorie, systeme, logiciel).

6.5.2.4 typeUtilisateur()

```
QString GestionnaireDialogue::typeUtilisateur (
    QString id,
    QString mdp )
```

Méthode qui permet d'identifier le type d'[Utilisateur](#) qui se connecte si il existe.

Parameters

<i>id</i>	L'identifiant de l' Utilisateur .
<i>mdp</i>	Le mot de passe de l' Utilisateur .

Returns

le nom du type d'utilisateur, null si l'utilisateur n'existe pas.

6.5.2.5 verificationMessage()

```
bool GestionnaireDialogue::verificationMessage (
    QString message )
```

Méthode permettant de vérifier que le message est bon.

Returns

true si le message est bon, false sinon.

The documentation for this class was generated from the following files:

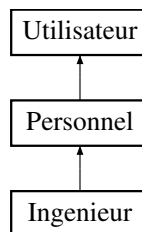
- include/gestionnairedialogue.h
- src/gestionnairedialogue.cpp

6.6 Ingenieur Class Reference

La classe [Ingenieur](#) permet de décrire un utilisateur appartenant au [Personnel](#) avec les droits ingénieur sur le programme EasyTicket.

```
#include <ingenieur.h>
```

Inheritance diagram for Ingenieur:



Public Member Functions

- [Ingenieur](#) ()
Constructeur par défaut de la classe.
- [Ingenieur](#) (QString [id](#), QString [nom](#), QString [prenom](#), QString [mdp](#), QString [mail](#))
Constructeur utilisant les données personnelles de l'ingénieur.
- [~Ingenieur](#) ()
Destructeur de la classe.
- bool [estUnIngenieur](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un Ingénieur.

Additional Inherited Members

6.6.1 Detailed Description

La classe [Ingenieur](#) permet de décrire un utilisateur appartenant au [Personnel](#) avec les droits ingénieur sur le programme EasyTicket.

Authors

Claire Kurth, Nicolas Robert, Victor Dallé

Version

1.1

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Ingenieur()

```
Ingenieur::Ingenieur (
    QString id,
    QString nom,
    QString prenom,
    QString mdp,
    QString mail )
```

Constructeur utilisant les données personnelles de l'ingénieur.

Parameters

<i>id</i>	L'identifiant de l'ingénieur.
<i>nom</i>	Le nom de l'ingénieur.
<i>prenom</i>	Le prénom de l'ingénieur.
<i>mdp</i>	Le mot de passe de l'ingénieur.
<i>mail</i>	Le mail de l'ingénieur.

6.6.3 Member Function Documentation

6.6.3.1 estUnIngenieur()

```
bool Ingenieur::estUnIngenieur ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un Ingénieur.

Returns

true

Reimplemented from [Utilisateur](#).

The documentation for this class was generated from the following files:

- include/ingenieur.h
- src/ingenieur.cpp

6.7 Logiciel Class Reference

La classe [Logiciel](#) permet de décrire un logiciel dans l'application EasyTicket.

```
#include <logiciel.h>
```

Public Member Functions

- [Logiciel](#) ()
Constructeur par défaut.
- [Logiciel](#) (int id_logiciel, QString nom)
Constructeur de la classe logiciel.
- QString [getNom](#) ()
Méthode permettant de récupérer le nom du [Logiciel](#).
- int [getID](#) ()
Méthode retournant l'id du logiciel.

6.7.1 Detailed Description

La classe [Logiciel](#) permet de décrire un logiciel dans l'application EasyTicket.

Authors

Nicolas Robert, Victor Dallé, Claire Kurth

Version

1.5

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Logiciel()

```
Logiciel::Logiciel (  
    int id_logiciel,  
    QString nom )
```

Constructeur de la classe logiciel.

Parameters

<i>id_logiciel</i>	l'identifiant du logiciel
<i>nom</i>	le nom du logiciel

6.7.3 Member Function Documentation

6.7.3.1 getID()

```
int Logiciel::getID ( )
```

Méthode retournant l'id du logiciel.

Returns

L'id du logiciel.

6.7.3.2 getNom()

```
QString Logiciel::getNom ( )
```

Méthode permettant de récupérer le nom du [Logiciel](#).

Returns

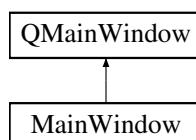
nom

The documentation for this class was generated from the following files:

- include/logiciel.h
- src/logiciel.cpp

6.8 MainWindow Class Reference

Inheritance diagram for MainWindow:



Public Member Functions

- [MainWindow](#) (QWidget *parent=nullptr)
Constructeur.
- void [setGestionnaireDialogue](#) ([GestionnaireDialogue](#) *g)
Méthode pour set le [GestionnaireDialogue](#).
- [~MainWindow](#) ()
Destructeur.

Protected Attributes

- QStackedWidget * [stack](#)
Champs contenant le StackedWidget qui stock les différentes Page de l'application.
- [GestionnaireDialogue](#) * [gestionnaire_dialogue](#)
Champs contenant le [GestionnaireDialogue](#).
- [PageLogin](#) * [page_login](#)
Champs contenant la [PageLogin](#).
- [PageAccueilClient](#) * [page_accueil_client](#)
Champs contenant la [PageAccueilClient](#).
- [PageAjoutTicket](#) * [page_ajout_ticket](#)
Champs contenant la [PageAjoutTicket](#).

6.8.1 Constructor & Destructor Documentation

6.8.1.1 MainWindow()

```
MainWindow::MainWindow (
    QWidget * parent = nullptr )
```

Constructeur.

Parameters

<i>*parent</i>	Pointeur sur le Widget Parent.
----------------	--------------------------------

6.8.2 Member Function Documentation

6.8.2.1 setGestionnaireDialogue()

```
void MainWindow::setGestionnaireDialogue (
    GestionnaireDialogue * g )
```

Méthode pour set le [GestionnaireDialogue](#).

Parameters

<i>g</i>	Pointeur sur le GestionnaireDialogue .
----------	--

The documentation for this class was generated from the following files:

- include/mainwindow.h
- src/mainwindow.cpp

6.9 Message Class Reference

La classe [Message](#) est la classe correspondant à un [Message](#) sur un [Ticket](#).

```
#include <message.h>
```

Public Member Functions

- [Message](#) ()
Le constructeur par défaut de la classe.
- [Message](#) (QString date_envoi, [Utilisateur](#) *user, [Ticket](#) *ticket, QString message, int id_message=-1)
Constructeur de message.
- int [getId_message](#) () const
Cette méthode permet de récupérer l'identifiant du message.
- void [setId_message](#) (const int &value)
Cette méthode permet de définir l'identifiant du message.
- QString [getDate_envoi](#) () const
Cette méthode permet de récupérer la date à laquelle le message a été envoyé.
- void [setDate_envoi](#) (QString value)
Cette méthode permet de définir la date à laquelle a été envoyé le message.
- [Utilisateur](#) * [getUser](#) () const
Cette méthode permet de récupérer l'[Utilisateur](#) auteur du message.
- void [setUser](#) ([Utilisateur](#) *&value)
Cette méthode permet de définir l'[Utilisateur](#) auteur du message.
- [Ticket](#) * [getTicket](#) () const
Cette méthode permet de récupérer le [Ticket](#) dans lequel se trouve le message.
- void [setTicket](#) ([Ticket](#) *value)
Cette méthode définit le [Ticket](#) dans lequel se trouve le message.
- QString [toString](#) ()
Cette méthode permet d'afficher les attributs de la classe pour le debug.
- QString [getMessage](#) () const
Cette méthode permet de récupérer le contenu d'un message.
- void [setMessage](#) (const QString &value)
Cette méthode permet de définir le contenu du message.
- [~Message](#) ()
Destructeur de la classe.

6.9.1 Detailed Description

La classe [Message](#) est la classe correspondant à un [Message](#) sur un [Ticket](#).

Authors

Nicolas Robert, Victor Dallé, Claire Kurth

Version

1.6

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Message()

```
Message::Message (
    QString date_envoi,
    Utilisateur * user,
    Ticket * ticket,
    QString message,
    int id_message = -1 )
```

Constructeur de message.

Parameters

<i>id_message</i>	L'id du message.
<i>date_envoi</i>	La date d'envoi du message.
<i>*user</i>	Pointeur sur l' Utilisateur à l'origine du message.
<i>*ticket</i>	Pointeur sur le Ticket rattaché au message.
<i>message</i>	Le message du Message .

6.9.3 Member Function Documentation

6.9.3.1 getDate_envoi()

```
QString Message::getDate_envoi ( ) const
```

Cette méthode permet de récupérer la date à laquelle le message a été envoyé.

Returns

la date d'envoi du message

6.9.3.2 getId_message()

```
int Message::getId_message ( ) const
```

Cette méthode permet de récupérer l'identifiant du message.

Returns

l'identifiant du message

6.9.3.3 getMessage()

```
QString Message::getMessage ( ) const
```

Cette méthode permet de récupérer le contenu d'un message.

Returns

le contenu du message.

6.9.3.4 getTicket()

```
Ticket * Message::getTicket ( ) const
```

Cette méthode permet de récupérer le [Ticket](#) dans lequel se trouve le message.

Returns

la classe [Ticket](#)

6.9.3.5 getUser()

```
Utilisateur * Message::getUser ( ) const
```

Cette méthode permet de récupérer l'[Utilisateur](#) auteur du message.

Returns

le nouvel [Utilisateur](#).

6.9.3.6 setDate_envoi()

```
void Message::setDate_envoi (
    QString value )
```

Cette méthode permet de définir la date à laquelle a été envoyé le message.

Parameters

<i>value</i>	la nouvelle date d'envoi
--------------	--------------------------

6.9.3.7 setId_message()

```
void Message::setId_message (  
    const int & value )
```

Cette méthode permet de définir l'identifiant du message.

Parameters

<i>value</i>	Le nouvel identifiant du message
--------------	----------------------------------

6.9.3.8 setMessage()

```
void Message::setMessage (  
    const QString & value )
```

Cette méthode permet définir le contenu du message.

Parameters

<i>value</i>	Le nouveau contenu du message.
--------------	--------------------------------

6.9.3.9 setTicket()

```
void Message::setTicket (  
    Ticket * value )
```

Cette méthode définit le [Ticket](#) dans lequel se trouve le message.

Parameters

<i>value</i>	Le nouveau Ticket
--------------	-----------------------------------

6.9.3.10 setUser()

```
void Message::setUser (
    Utilisateur *& value )
```

Cette méthode permet de définir l'[Utilisateur](#) auteur du message.

Parameters

<i>value</i>	Le nouvel Utilisateur .
--------------	---

6.9.3.11 toString()

```
QString Message::toString ( )
```

Cette méthode permet d'afficher les attributs de la classe pour le debug.

Returns

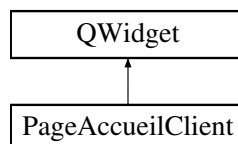
String avec les infos de la classe.

The documentation for this class was generated from the following files:

- include/message.h
- src/message.cpp

6.10 PageAccueilClient Class Reference

Inheritance diagram for PageAccueilClient:



Public Member Functions

- [PageAccueilClient](#) (QWidget *parent=nullptr, [GestionnaireDialogue](#) *gestionnaire_dialogue=nullptr)
Constructeur par défaut.
- void [setClient](#) ([Client](#) client)
Méthode permettant de set le [Client](#) actuellement connecté.
- [~PageAccueilClient](#) ()
Destructeur.

6.10.1 Constructor & Destructor Documentation

6.10.1.1 PageAccueilClient()

```
PageAccueilClient::PageAccueilClient (
    QWidget * parent = nullptr,
    GestionnaireDialogue * gestionnaire_dialogue = nullptr ) [explicit]
```

Constructeur par défaut.

Parameters

<i>*parent</i>	Widget parent.
<i>*gestionnaire_dialogue</i>	le GestionnaireDialogue de l'application

6.10.2 Member Function Documentation

6.10.2.1 setClient()

```
void PageAccueilClient::setClient (
    Client client )
```

Méthode permettant de set le [Client](#) actuellement connecté.

Parameters

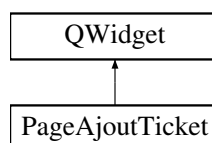
<i>client</i>	le Client connecté.
---------------	-------------------------------------

The documentation for this class was generated from the following files:

- include/pageaccueilclient.h
- src/pageaccueilclient.cpp

6.11 PageAjoutTicket Class Reference

Inheritance diagram for PageAjoutTicket:



Public Member Functions

- [PageAjoutTicket](#) (QWidget *parent=nullptr, [GestionnaireDialogue](#) *gestionnaire_dialogue=nullptr)
Constructeur.
- void [setClient](#) ([Client](#) client)
Méthode permettant de set le [Client](#) actuellement connecté.

6.11.1 Constructor & Destructor Documentation

6.11.1.1 PageAjoutTicket()

```
PageAjoutTicket::PageAjoutTicket (
    QWidget * parent = nullptr,
    GestionnaireDialogue * gestionnaire_dialogue = nullptr ) [explicit]
```

Constructeur.

Parameters

<i>parent</i>	Widget parent.
<i>*gestionnaire_dialogue</i>	le GestionnaireDialogue de l'application

6.11.2 Member Function Documentation

6.11.2.1 setClient()

```
void PageAjoutTicket::setClient (
    Client client )
```

Méthode permettant de set le [Client](#) actuellement connecté.

Parameters

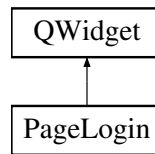
<i>client</i>	le Client connecté.
---------------	-------------------------------------

The documentation for this class was generated from the following files:

- include/pageajoutticket.h
- src/pageajoutticket.cpp

6.12 PageLogin Class Reference

Inheritance diagram for PageLogin:



Public Slots

- void [handle_validation](#) ()
Cette méthode est appelée lors d'un click sur le bouton validation_bouton. Elle permet la connexion de l'[Utilisateur](#) si celui ci existe.

Public Member Functions

- [PageLogin](#) (QWidget *parent=nullptr, [GestionnaireDialogue](#) *gestionnaire_dialogue=nullptr)
Constructeur.
- [~PageLogin](#) ()
Destructeur.

6.12.1 Constructor & Destructor Documentation

6.12.1.1 PageLogin()

```

PageLogin::PageLogin (
    QWidget * parent = nullptr,
    GestionnaireDialogue * gestionnaire_dialogue = nullptr ) [explicit]
  
```

Constructeur.

Parameters

<i>parent</i>	Widget parent.
<i>*gestionnaire_dialogue</i>	le GestionnaireDialogue de l'application

The documentation for this class was generated from the following files:

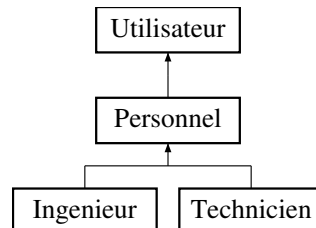
- include/pagelogin.h
- src/pagelogin.cpp

6.13 Personnel Class Reference

La classe [Personnel](#) est une classe abstraite qui permet de décrire l'ensemble des [Utilisateur](#) appartenant au personnel dans l'application EasyTicket.

```
#include <personnel.h>
```

Inheritance diagram for Personnel:



Public Member Functions

- [Personnel](#) ()
Constructeur par défaut de la classe.
- [Personnel](#) (QString `id`, QString `nom`, QString `prenom`, QString `mdp`, QString `mail`)
Constructeur utilisant les données personnelles du personnel.
- virtual [~Personnel](#) ()
Destructeur virtuel pour rendre la classe [Personnel](#) abstraite.
- virtual bool [estUnPersonnel](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Personnel](#).

Additional Inherited Members

6.13.1 Detailed Description

La classe [Personnel](#) est une classe abstraite qui permet de décrire l'ensemble des [Utilisateur](#) appartenant au personnel dans l'application EasyTicket.

Elle sert de classe mère aux autres classes comme [Ingenieur](#) et [Technicien](#).

Authors

Claire Kurth, Nicolas Robert, Victor Dallé

Version

1.1

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Personnel()

```
Personnel::Personnel (
    QString id,
    QString nom,
    QString prenom,
    QString mdp,
    QString mail )
```

Constructeur utilisant les données personnelles du personnel.

Parameters

<i>id</i>	L'identifiant du personnel.
<i>nom</i>	Le nom du personnel.
<i>prenom</i>	Le prénom du personnel.
<i>mdp</i>	Le mot de passe du personnel.
<i>mail</i>	Le mail du personnel.

6.13.3 Member Function Documentation

6.13.3.1 estUnPersonnel()

```
bool Personnel::estUnPersonnel ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Personnel](#).

Returns

true

Reimplemented from [Utilisateur](#).

The documentation for this class was generated from the following files:

- include/personnel.h
- src/personnel.cpp

6.14 Systeme Class Reference

La classe [Systeme](#) permet de décrire un système d'exploitation dans l'application EasyTicket.

```
#include <systeme.h>
```

Public Member Functions

- [Systeme](#) ()
Constructeur par défaut.
- [Systeme](#) (int id_systeme, QString nom)
Constructeur de la classe [Systeme](#).
- int [getId_systeme](#) () const
Cette méthode permet de récupérer l'identifiant du système.
- void [setId_systeme](#) (const int &value)
Cette méthode permet de définir l'identifiant du système.
- QString [getNom](#) () const
Cette méthode permet de récupérer le nom du système.
- void [setNom](#) (const QString &value)
Cette méthode permet de définir le nom du système.
- [~Systeme](#) ()
Destructeur de la classe.

6.14.1 Detailed Description

La classe [Systeme](#) permet de décrire un système d'exploitation dans l'application EasyTicket.

Authors

Nicolas Robert, Victor Dallé, Claire Kurth

Version

1.1

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Systeme()

```
Systeme::Systeme (
    int id_systeme,
    QString nom )
```

Constructeur de la classe [Systeme](#).

Parameters

<i>id_systeme</i>	L'identifiant du système d'exploitation
<i>nom</i>	Le nom du système d'exploitation

6.14.3 Member Function Documentation

6.14.3.1 getId_systeme()

```
int Systeme::getId_systeme ( ) const
```

Cette méthode permet de récupérer l'identifiant du système.

Returns

l'identifiant du système

6.14.3.2 getNom()

```
QString Systeme::getNom ( ) const
```

Cette méthode permet de récupérer le nom du système.

Returns

le nom du système

6.14.3.3 setId_systeme()

```
void Systeme::setId_systeme (
    const int & value )
```

Cette méthode permet de définir l'identifiant du système.

Parameters

<i>value</i>	l'identifiant du système
--------------	--------------------------

6.14.3.4 setNom()

```
void Systeme::setNom (
    const QString & value )
```

Cette méthode permet de définir le nom du système.

Parameters

<i>value</i>	le nouveau nom du système
--------------	---------------------------

The documentation for this class was generated from the following files:

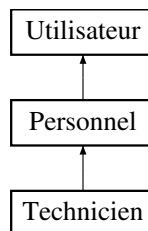
- include/systeme.h
- src/systeme.cpp

6.15 Technicien Class Reference

La classe [Technicien](#) permet de décrire un utilisateur appartenant au [Personnel](#) avec les droits technicien sur le programme EasyTicket.

```
#include <technicien.h>
```

Inheritance diagram for Technicien:



Public Member Functions

- [Technicien](#) ()
Constructeur par défaut de la classe.
- [Technicien](#) (QString [id](#), QString [nom](#), QString [prenom](#), QString [mdp](#), QString [mail](#), std::vector< [Categorie](#) * > categories)
Constructeur utilisant les données personnelles du technicien.
- [~Technicien](#) ()
Destructeur de la classe.
- bool [estUnTechnicien](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Technicien](#).

Additional Inherited Members

6.15.1 Detailed Description

La classe [Technicien](#) permet de décrire un utilisateur appartenant au [Personnel](#) avec les droits technicien sur le programme EasyTicket.

Authors

Claire Kurth, Nicolas Robert, Victor Dallé

Version

1.1

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Technicien()

```
Technicien::Technicien (
    QString id,
    QString nom,
    QString prenom,
    QString mdp,
    QString mail,
    std::vector< Categorie * > categories )
```

Constructeur utilisant les données personnelles du technicien.

Parameters

<i>id</i>	L'identifiant du technicien.
<i>nom</i>	Le nom du technicien.
<i>prenom</i>	Le prénom du technicien.
<i>mdp</i>	Le mot de passe du technicien.
<i>mail</i>	Le mail du technicien.
<i>categories</i>	La liste de catégories maîtrisées par le technicien.

6.15.3 Member Function Documentation

6.15.3.1 estUnTechnicien()

```
bool Technicien::estUnTechnicien ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Technicien](#).

Returns

true

Reimplemented from [Utilisateur](#).

The documentation for this class was generated from the following files:

- include/technicien.h
- src/technicien.cpp

6.16 Ticket Class Reference

La classe [Ticket](#) permet de décrire un ticket dans l'application EasyTicket.

```
#include <ticket.h>
```

Public Member Functions

- [Ticket](#) ()
Constructeur par défaut.
- [Ticket](#) (QString date_creation, [Categorie](#) categorie, [Client](#) auteur, int id_ticket=-1)
Constructeur de la classe [Ticket](#) avec les informations obligatoires.
- int [getId_ticket](#) () const
Cette méthode permet de récupérer l'identifiant du [Ticket](#).
- void [setId_ticket](#) (const int &value)
Cette méthode permet de définir l'identifiant du [Ticket](#).
- QString [getDate_creation](#) () const
Cette méthode permet de récupérer la date de création du [Ticket](#).
- void [setDate_creation](#) (QString value)
Cette méthode permet de définir la date de création du [Ticket](#).
- QString [getDate_fermeture](#) () const
Cette méthode permet de récupérer la date de fermeture du [Ticket](#).
- void [setDate_fermeture](#) (QString value)
Cette méthode permet de définir la date de fermeture du [Ticket](#).
- [Systeme](#) [getSysteme](#) () const
Cette méthode permet de récupérer le système d'exploitation concerné par le [Ticket](#).
- void [setSysteme](#) (const [Systeme](#) &value)
Cette méthode permet de définir le [Systeme](#) concerné par le [Ticket](#).
- [Logiciel](#) [getLogiciel](#) () const
Cette méthode permet récupérer le [Logiciel](#) concerné par le [Ticket](#).
- void [setLogiciel](#) (const [Logiciel](#) &value)
Cette méthode permet définir le [Logiciel](#) concerné par le [Ticket](#).
- std::vector< [Message](#) * > [getListeMessages](#) () const
Cette méthode permet de récupérer la liste des [Message](#) concerné par le [Ticket](#).
- void [addMessage](#) ([Message](#) &message)
Cette méthode permet d'ajouter un [Message](#) concerné par le [Ticket](#).
- void [setListeMessages](#) (const std::vector< [Message](#) * > &value)
Cette méthode permet de définir la liste des [@Message](#) concerné par le [Ticket](#).
- void [setClient](#) ([Client](#) &value)
Cette méthode permet de définir le [Client](#) auteur du [Ticket](#).
- [Client](#) [getClient](#) ()
Cette méthode permet de récupérer le [Client](#) auteur du [Ticket](#).
- void [setPersonnel](#) ([Personnel](#) &value)
Cette méthode permet de définir le [Personnel](#) qui traite le [Ticket](#).
- [Personnel](#) [getPersonnel](#) ()
Cette méthode permet de récupérer le [Personnel](#) qui traite le [Ticket](#).
- QString [toString](#) ()
Cette méthode permet d'afficher les attributs de la classe pour le debug.
- [Categorie](#) [getCategorie](#) ()
- [~Ticket](#) ()
Destructeur de la classe.

6.16.1 Detailed Description

La classe [Ticket](#) permet de décrire un ticket dans l'application EasyTicket.

Authors

Nicolas Robert, Victor Dallé, Claire Kurth

Version

1.6

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Ticket()

```
Ticket::Ticket (
    QString date_creation,
    Categorie categorie,
    Client auteur,
    int id_ticket = -1 )
```

Constructeur de la classe [Ticket](#) avec les informations obligatoires.

Parameters

<i>id_ticket</i>	l'identifiant du ticket
<i>date_creation</i>	la date de création du ticket
<i>categorie</i>	la Categorie concernée par le ticket
<i>auteur</i>	le Client auteur du ticket

6.16.3 Member Function Documentation

6.16.3.1 addMessage()

```
void Ticket::addMessage (
    Message & message )
```

Cette méthode permet d'ajouter un [Message](#) concerné par le [Ticket](#).

Parameters

<i>value</i>	La nouvelle liste de Message
--------------	--

6.16.3.2 getClient()

```
Client Ticket::getClient ( )
```

Cette méthode permet de récupérer le [Client](#) auteur du [Ticket](#).

Returns

Le [Client](#) auteur du [Ticket](#)

6.16.3.3 getDate_creation()

```
QString Ticket::getDate_creation ( ) const
```

Cette méthode permet de récupérer la date de création du [Ticket](#).

Returns

La date de création du ticket

6.16.3.4 getDate_fermeture()

```
QString Ticket::getDate_fermeture ( ) const
```

Cette méthode permet de récupérer la date de fermeture du [Ticket](#).

Returns

La date de fermeture du ticket

6.16.3.5 getId_ticket()

```
int Ticket::getId_ticket ( ) const
```

Cette méthode permet de récupérer l'identifiant du [Ticket](#).

Returns

L'identifiant du ticket

6.16.3.6 getListeMessages()

```
std::vector< Message * > Ticket::getListeMessages ( ) const
```

Cette méthode permet de récupérer la liste des [Message](#) concerné par le [Ticket](#).

Returns

La liste de [Message](#) concerné par le ticket

6.16.3.7 getLogiciel()

```
Logiciel Ticket::getLogiciel ( ) const
```

Cette méthode permet récupérer le [Logiciel](#) concerné par le [Ticket](#).

Returns

Le [Logiciel](#) concerné par le [Ticket](#)

6.16.3.8 getPersonnel()

```
Personnel Ticket::getPersonnel ( )
```

Cette méthode permet de récupérer le [Personnel](#) qui traite le [Ticket](#).

Returns

Le Personel traitant le [Ticket](#)

6.16.3.9 getSysteme()

```
Systeme Ticket::getSysteme ( ) const
```

Cette méthode permet de récupérer le système d'exploitation concerné par le [Ticket](#).

Returns

La date de fermeture du ticket

6.16.3.10 setClient()

```
void Ticket::setClient (
    Client & value )
```

Cette méthode permet de définir le [Client](#) auteur du [Ticket](#).

Parameters

<i>value</i>	Le Client auteur du Ticket
--------------	--

6.16.3.11 setDate_creation()

```
void Ticket::setDate_creation (
    QString value )
```

Cette méthode permet de définir la date de création du [Ticket](#).

Parameters

<i>value</i>	La nouvelle date de création du Ticket
--------------	--

6.16.3.12 setDate_fermeture()

```
void Ticket::setDate_fermeture (
    QString value )
```

Cette méthode permet de définir la date de fermeture du [Ticket](#).

Parameters

<i>value</i>	La nouvelle date de fermeture du Ticket
--------------	---

6.16.3.13 setId_ticket()

```
void Ticket::setId_ticket (
    const int & value )
```

Cette méthode permet de définir l'identifiant du [Ticket](#).

Parameters

<i>value</i>	Le nouvel identifiant du ticket
--------------	---------------------------------

6.16.3.14 setListeMessages()

```
void Ticket::setListeMessages (
    const std::vector< Message * > & value )
```

Cette méthode permet de définir la liste des @Message concerné par le Ticket.

Parameters

<i>value</i>	La nouvelle liste de Message
--------------	------------------------------

6.16.3.15 setLogiciel()

```
void Ticket::setLogiciel (
    const Logiciel & value )
```

Cette méthode permet définir le Logiciel concerné par le Ticket.

Parameters

<i>value</i>	Le nouveau Logiciel concerné par le Ticket
--------------	--

6.16.3.16 setPersonnel()

```
void Ticket::setPersonnel (
    Personnel & value )
```

Cette méthode permet de définir le Personnel qui traite le Ticket.

Parameters

<i>value</i>	Le nouveau Personnel
--------------	----------------------

6.16.3.17 setSysteme()

```
void Ticket::setSysteme (
    const Systeme & value )
```

Cette méthode permet de définir le Systeme concerné par le Ticket.

Parameters

value	Le nouveau Système concerné par le Ticket
-------	---

The documentation for this class was generated from the following files:

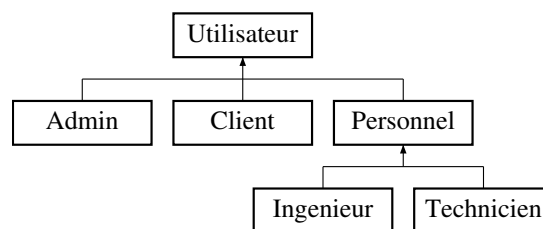
- include/ticket.h
- src/ticket.cpp

6.17 Utilisateur Class Reference

La classe [Utilisateur](#) est une classe abstraite qui permet de décrire l'ensemble des utilisateurs dans l'application EasyTicket.

```
#include <utilisateur.h>
```

Inheritance diagram for Utilisateur:



Public Member Functions

- [Utilisateur](#) ()
Constructeur par défaut de la classe.
- [Utilisateur](#) (QString [id](#), QString [nom](#), QString [prenom](#), QString [mdp](#), QString [mail](#))
Constructeur utilisant les données personnelles de l'utilisateur.
- virtual [~Utilisateur](#) ()
Destructeur virtuel pour rendre la classe abstraite.
- QString [getID](#) () const
Cette méthode permet de récupérer l'identifiant de l'utilisateur.
- void [setID](#) (const QString &value)
Cette méthode permet de définir l'identifiant de l'utilisateur.
- QString [getNom](#) () const
Cette méthode permet de récupérer le nom de l'utilisateur.
- void [setNom](#) (const QString &value)
Cette méthode permet de définir le nom de l'utilisateur.
- QString [getPrenom](#) () const
Cette méthode permet de récupérer le prénom de l'utilisateur.
- void [setPrenom](#) (const QString &value)
Cette méthode permet de définir le prénom de l'utilisateur.
- QString [getMDP](#) () const
Cette méthode permet de récupérer le mot de passe de l'utilisateur.

- void [setMDP](#) (const QString &value)
Cette méthode permet de définir le mot de passe de l'utilisateur.
- QString [getMail](#) () const
Cette méthode permet de récupérer le mail de l'utilisateur.
- void [setMail](#) (const QString &value)
Cette méthode permet de définir le mail de l'utilisateur.
- virtual bool [estUnClient](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Client](#).
- virtual bool [estUnPersonnel](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Personnel](#).
- virtual bool [estUnAdmin](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Admin](#).
- virtual bool [estUnTechnicien](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Technicien](#).
- virtual bool [estUnIngenieur](#) ()
Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Ingenieur](#).
- QString [toString](#) ()
Cette méthode permet d'afficher les attributs de la classe pour le debug.

Protected Attributes

- QString [id](#)
Champs contenant l'identifiant de l'utilisateur.
- QString [nom](#)
Champs contenant le nom de l'utilisateur.
- QString [prenom](#)
Champs contenant le prénom de l'utilisateur.
- QString [mdp](#)
Champs contenant le mot de passe de l'utilisateur.
- QString [mail](#)
Champs contenant l'adresse e-mail de l'utilisateur.
- std::vector< [Ticket](#) * > [liste_tickets](#)
Champs contenant la liste des tickets associée à l'utilisateur.

6.17.1 Detailed Description

La classe [Utilisateur](#) est une classe abstraite qui permet de décrire l'ensemble des utilisateurs dans l'application EasyTicket.

Elle sert de classe mère aux autres classes comme [Client](#), [Personnel](#) et [Admin](#).

Authors

Claire Kurth, Nicolas Robert, Victor Dallé

Version

1.1

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Utilisateur()

```
Utilisateur::Utilisateur (
    QString id,
    QString nom,
    QString prenom,
    QString mdp,
    QString mail )
```

Constructeur utilisant les données personnelles de l'utilisateur.

Parameters

<i>id</i>	L'identifiant de l'utilisateur.
<i>nom</i>	Le nom de l'utilisateur.
<i>prenom</i>	Le prénom de l'utilisateur.
<i>mdp</i>	Le mot de passe de l'utilisateur.
<i>mail</i>	Le mail de l'utilisateur.

6.17.3 Member Function Documentation

6.17.3.1 estUnAdmin()

```
bool Utilisateur::estUnAdmin ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Admin](#).

Returns

false

Reimplemented in [Admin](#).

6.17.3.2 estUnClient()

```
bool Utilisateur::estUnClient ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Client](#).

Returns

false

Reimplemented in [Client](#).

6.17.3.3 estUnIngenieur()

```
bool Utilisateur::estUnIngenieur ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Ingenieur](#).

Returns

false

Reimplemented in [Ingenieur](#).

6.17.3.4 estUnPersonnel()

```
bool Utilisateur::estUnPersonnel ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Personnel](#).

Returns

false

Reimplemented in [Personnel](#).

6.17.3.5 estUnTechnicien()

```
bool Utilisateur::estUnTechnicien ( ) [virtual]
```

Cette méthode permet de vérifier si l'[Utilisateur](#) est un [Technicien](#).

Returns

false

Reimplemented in [Technicien](#).

6.17.3.6 getID()

```
QString Utilisateur::getID ( ) const
```

Cette méthode permet de récupérer l'identifiant de l'utilisateur.

Returns

L'identifiant de l'utilisateur.

6.17.3.7 getMail()

```
QString Utilisateur::getMail ( ) const
```

Cette méthode permet de récupérer le mail de l'utilisateur.

Returns

Le mail de l'utilisateur.

6.17.3.8 getMDP()

```
QString Utilisateur::getMDP ( ) const
```

Cette méthode permet de récupérer le mot de passe de l'utilisateur.

Returns

Le mot de passe de l'utilisateur.

6.17.3.9 getNom()

```
QString Utilisateur::getNom ( ) const
```

Cette méthode permet de récupérer le nom de l'utilisateur.

Returns

Le nom de l'utilisateur.

6.17.3.10 getPrenom()

```
QString Utilisateur::getPrenom ( ) const
```

Cette méthode permet de récupérer le prénom de l'utilisateur.

Returns

Le prénom de l'utilisateur.

6.17.3.11 setID()

```
void Utilisateur::setID (
    const QString & value )
```

Cette méthode permet de définir l'identifiant de l'utilisateur.

Parameters

<i>value</i>	Le nouvel identifiant de l'utilisateur.
--------------	---

6.17.3.12 setMail()

```
void Utilisateur::setMail (
    const QString & value )
```

Cette méthode permet de définir le mail de l'utilisateur.

Parameters

<i>value</i>	Le nouveau mail de l'utilisateur.
--------------	-----------------------------------

6.17.3.13 setMDP()

```
void Utilisateur::setMDP (
    const QString & value )
```

Cette méthode permet de définir le mot de passe de l'utilisateur.

Parameters

<i>value</i>	Le nouveau mot de passe de l'utilisateur.
--------------	---

6.17.3.14 setNom()

```
void Utilisateur::setNom (
    const QString & value )
```

Cette méthode permet de définir le nom de l'utilisateur.

Parameters

<i>value</i>	Le nouveau nom de l'utilisateur.
--------------	----------------------------------

6.17.3.15 setPrenom()

```
void Utilisateur::setPrenom (
    const QString & value )
```

Cette méthode permet de définir le prénom de l'utilisateur.

Parameters

<i>value</i>	Le nouveau prénom de l'utilisateur.
--------------	-------------------------------------

The documentation for this class was generated from the following files:

- include/utilisateur.h
- src/utilisateur.cpp

Index

- addMessage
 - Ticket, [41](#)
- Admin, [11](#)
 - Admin, [12](#)
 - estUnAdmin, [12](#)
- authentification
 - GestionnaireDialogue, [19](#)
- authentifieur
 - GestionnaireBDD, [17](#)
- Categorie, [12](#)
 - Categorie, [13](#)
 - getID, [13](#)
 - getNom, [14](#)
- Client, [14](#)
 - Client, [15](#)
 - estUnClient, [15](#)
- enregistrer_ticket
 - GestionnaireBDD, [17](#)
 - GestionnaireDialogue, [19](#)
- estUnAdmin
 - Admin, [12](#)
 - Utilisateur, [48](#)
- estUnClient
 - Client, [15](#)
 - Utilisateur, [48](#)
- estUnIngenieur
 - Ingenieur, [22](#)
 - Utilisateur, [48](#)
- estUnPersonnel
 - Personnel, [35](#)
 - Utilisateur, [49](#)
- estUnTechnicien
 - Technicien, [39](#)
 - Utilisateur, [49](#)
- GestionnaireBDD, [16](#)
 - authentifieur, [17](#)
 - enregistrer_ticket, [17](#)
 - select, [17](#)
 - setComboBox, [18](#)
 - type_utilisateur, [18](#)
- GestionnaireDialogue, [18](#)
 - authentification, [19](#)
 - enregistrer_ticket, [19](#)
 - setComboBox, [20](#)
 - typeUtilisateur, [20](#)
 - verificationMessage, [20](#)
- getClient
 - Ticket, [42](#)
- getDate_creation
 - Ticket, [42](#)
- getDate_envoi
 - Message, [27](#)
- getDate_fermeture
 - Ticket, [42](#)
- getID
 - Categorie, [13](#)
 - Logiciel, [24](#)
 - Utilisateur, [49](#)
- getId_message
 - Message, [27](#)
- getId_systeme
 - Systeme, [37](#)
- getId_ticket
 - Ticket, [42](#)
- getListeMessages
 - Ticket, [42](#)
- getLogiciel
 - Ticket, [43](#)
- getMail
 - Utilisateur, [49](#)
- getMDP
 - Utilisateur, [50](#)
- getMessage
 - Message, [28](#)
- getNom
 - Categorie, [14](#)
 - Logiciel, [24](#)
 - Systeme, [37](#)
 - Utilisateur, [50](#)
- getPersonnel
 - Ticket, [43](#)
- getPrenom
 - Utilisateur, [50](#)
- getSysteme
 - Ticket, [43](#)
- getTicket
 - Message, [28](#)
- getUser
 - Message, [28](#)
- Ingenieur, [21](#)
 - estUnIngenieur, [22](#)
 - Ingenieur, [22](#)
- Logiciel, [23](#)
 - getID, [24](#)
 - getNom, [24](#)

- Logiciel, 23
- MainWindow, 24
 - MainWindow, 25
 - setGestionnaireDialogue, 25
- Message, 26
 - getDate_envoi, 27
 - getId_message, 27
 - getMessage, 28
 - getTicket, 28
 - getUser, 28
 - Message, 27
 - setDate_envoi, 28
 - setId_message, 29
 - setMessage, 29
 - setTicket, 29
 - setUser, 29
 - toString, 30
- PageAccueilClient, 30
 - PageAccueilClient, 31
 - setClient, 31
- PageAjoutTicket, 31
 - PageAjoutTicket, 32
 - setClient, 32
- PageLogin, 33
 - PageLogin, 33
- Personnel, 34
 - estUnPersonnel, 35
 - Personnel, 34
- select
 - GestionnaireBDD, 17
- setClient
 - PageAccueilClient, 31
 - PageAjoutTicket, 32
 - Ticket, 43
- setComboBox
 - GestionnaireBDD, 18
 - GestionnaireDialogue, 20
- setDate_creation
 - Ticket, 44
- setDate_envoi
 - Message, 28
- setDate_fermeture
 - Ticket, 44
- setGestionnaireDialogue
 - MainWindow, 25
- setId
 - Utilisateur, 50
- setId_message
 - Message, 29
- setId_systeme
 - Systeme, 37
- setId_ticket
 - Ticket, 44
- setListeMessages
 - Ticket, 44
- setLogiciel
- Ticket, 45
 - setMail
 - Utilisateur, 51
 - setMDP
 - Utilisateur, 51
 - setMessage
 - Message, 29
 - setNom
 - Systeme, 37
 - Utilisateur, 51
 - setPersonnel
 - Ticket, 45
 - setPrenom
 - Utilisateur, 51
 - setSysteme
 - Ticket, 45
 - setTicket
 - Message, 29
 - setUser
 - Message, 29
 - Systeme, 35
 - getId_systeme, 37
 - getNom, 37
 - setId_systeme, 37
 - setNom, 37
 - Systeme, 36
- Technicien, 38
 - estUnTechnicien, 39
 - Technicien, 39
- Ticket, 40
 - addMessage, 41
 - getClient, 42
 - getDate_creation, 42
 - getDate_fermeture, 42
 - getId_ticket, 42
 - getListeMessages, 42
 - getLogiciel, 43
 - getPersonnel, 43
 - getSysteme, 43
 - setClient, 43
 - setDate_creation, 44
 - setDate_fermeture, 44
 - setId_ticket, 44
 - setListeMessages, 44
 - setLogiciel, 45
 - setPersonnel, 45
 - setSysteme, 45
 - Ticket, 41
- toString
 - Message, 30
- type_utilisateur
 - GestionnaireBDD, 18
- typeUtilisateur
 - GestionnaireDialogue, 20
- Ui, 9
- Utilisateur, 46
 - estUnAdmin, 48

- estUnClient, [48](#)
- estUnIngenieur, [48](#)
- estUnPersonnel, [49](#)
- estUnTechnicien, [49](#)
- getID, [49](#)
- getMail, [49](#)
- getMDP, [50](#)
- getNom, [50](#)
- getPrenom, [50](#)
- setID, [50](#)
- setMail, [51](#)
- setMDP, [51](#)
- setNom, [51](#)
- setPrenom, [51](#)
- Utilisateur, [48](#)

verificationMessage

- GestionnaireDialogue, [20](#)