|  |  |  |
| --- | --- | --- |
| **BTS2 TP #4** | **POOkédex** | **2020/21** |

Ce TP propose de réaliser un Pokédex en utilisant le framework Qt. Un Pokédex est un appareil qui accompagne tout bon dresseur de Pokémon et qui liste les Pokémons et leurs informations.

Notre application émulera cet appareil et s’appuiera sur les webservices pokeapi.co pour récupérer informations sur les Pokémon et leurs images.

|  |  |  |
| --- | --- | --- |
| Sprite du Pokémon  Pokémon +/- 10  Pokémon précédent /suivant  Validation  Saisie d’un ID |  | Nom du Pokémon  Types du Pokémon  Evolutions du Pokémon  Retour au Pokémon précédent |

Par défaut, l’application affichera la fiche de Pikachu (25).

L’application affichera les données suivantes pour chaque Pokémon :

* Son sprite
* Son nom en anglais avec une majuscule
* Son type entre parenthèses
* Son ID
* Ses évolutions (s’il y en a)

Elle permettra d’accéder à d’autres fiches de Pokémon via les actions suivantes :

* Saisie d’un ID et validation avec le bouton bleu de validation ou appuie sur la touche « Entrée »
* Clic sur la flèche directionnelle :
  + Gauche/Droite décrémente/incrémente l’ID de 1
  + Haut/Bas incrémente/décrémente l’ID de 10
* Clic sur un bouton bleu d’évolution
* Clic sur le bouton jaune de retour au Pokémon suivant

Pour optimiser l’ergonomie de l’application, on s’attachera à afficher des textes d’assistance (tooltip) au survol des boutons et à changer le curseur en fonction de la disponibilité ou pas du bouton.

# Partie 1 - Etude du modèle de données

Consulter le site internet de l’API pokeapi.co, étudiez les différents services proposés et définissez les services nécessaires pour récupérer les informations suivantes :

* Nom du Pokémon en Anglais
* ID du Pokémon
* URL du sprite (image) du Pokémon
* Noms des types du Pokémon
* Noms des évolutions du Pokémon

# Partie 2 - Visuel du Pokédex

Ajouter le fichier pokedex\_bg.png comme ressource du projet avec le prefix « /pokeRessources ».

Dans le constructeur de la fenêtre principale, ajouter le code suivant pour ajouter le visuel du Pokédex.

setFixedSize(QSize(800, 542));

QPixmap bkgnd(":/pokeRessources/pokedex\_bg.png");

bkgnd = bkgnd.scaled(this->size(), Qt::KeepAspectRatio);

QPalette palette;

palette.setBrush(QPalette::Background, bkgnd);

setPalette(palette);

# Partie 3 - Premier appel de web service

L’appel des web services de la pokeapi.co se passera en deux étapes :

1. L’appel en lui-même
2. La gestion de la réponse

## Appel du web service

Ajouter le module Network à votre projet Qt en ajoutant la ligne suivante au début de votre fichier .pro :

QT += network

Pour contacter un web service, on utilisera un objet QNetworkAccessManager.

Cet objet contient une méthode get(QNetworkRequest(Qurl("url"))) qui permet d’envoyer une requête HTTP de type GET.

Au retour du serveur, l’objet émet un signal finished(QNetworkReply \*) qu’on connectera à un slot créé par nos soins.

La documentation : <https://doc.qt.io/qt-5/qnetworkaccessmanager.html>

1. Mettre en place une méthode getPokemon(int) dans la fenêtre principale qui prend en argument un ID et appelle le web service [https://pokeapi.co/api/v2/pokemon/{id}](https://pokeapi.co/api/v2/pokemon/%7bid%7d) .
2. Créer le slot managePokemonReply(QNetworkReply \*) et vérifier qu’il est bien appeler en utilisant qDebug()

## Gestion de la réponse

Le slot managePokemonReply(QNetworkReply \*) reçoit un pointeur sur un objet QNetworkReply qui possède une méthode readAll() qui permet de récupérer la réponse du serveur.

Le format de réponse choisi par pokeApi est le JSON. On va donc utiliser un objet qui nous permet de parcourir cette réponse : QJsonDocument.

QString source = reply->readAll();

QJsonObject pokeData;

QJsonDocument doc = QJsonDocument::fromJson(source.toUtf8());

if(!doc.isNull()) {

if(doc.isObject()) {

pokeData = doc.object();

// Travailler à partir d’ici avec l’objet pokeData

}

else {

qDebug() << "Document is not an object" << endl;

}

}

1. Implémenter le code ci-dessus dans le slot et récupérer le nom du Pokémon et l’afficher dans un label sur le Pokédex.
2. Récupérer l’URL du sprite du Pokémon

# Partie 4 - Téléchargement d’une image

Le téléchargement d’une image est une requête HTTP de type GET, le traitement de la réponse diffère puisqu’il faut recréer une image à partir de la réponse.

Voici le code pour créer une image à partir d’une réponse de type QNetworkReply et l’afficher dans un label.

QByteArray imgData = reply->readAll();

QPixmap pixmap;

pixmap.loadFromData(imgData);

if (!pixmap.isNull()) {

label->clear();

label->setPixmap(pixmap.scaled(label->size(), Qt::KeepAspectRatio));

}

1. Ajouter un label sur votre Pokédex pour accueillir le sprite.
2. Adapter le code ci-dessus à votre programme pour afficher le sprite du Pokémon dont on vient de recevoir les informations.

# Partie 5 - Terminer l’application

A partir d’ici, vous avez tous les éléments techniques pour réaliser complètement l’application.

1. Commencer par implémenter la fonctionnalité la plus « simple » : la recherche et l’affichage d’un Pokémon à partir de son ID.
2. Continuer en implémentant complètement votre IHM
3. Préparer les appels aux webservices complémentaires (évolutions…) et la gestion des réponses
4. Connecter tous les éléments
5. Penser aux modifications de l’IHM en fonction des réponses : inactivation de boutons, changement de curseur…

# Partie 7 - Bonus

* Afficher une barre de progression à chaque téléchargement.