SAE S5.01 - Apprentis

Descriptifs des défis de l'application Déf'IUT

Théo Challon, Liam Le Ny,
Ervan Malivet, Damien Masson, Augustin Pasquier

Groupe 3A

Vannes iut Université Bretagne Sud

iutvannes.fr

Table des matières

Introduction

Contexte

Ce document réalisé dans le cadre du BUT Informatique présente les descriptifs des défis visant à être intégrés sur la plateforme en ligne “Déf’IUT”. Pour rappel, cette plateforme a pour objectif de recueillir des défis informatiques qui seront résolus par les étudiants de première année.

Défis

Chaque défi se voit attribuer :

- Un **objectif**, correspondant à la consigne du défi sur la plateforme
- Une **difficulté**, sur une échelle de 1 à 5. Chaque difficulté a été justifiée dans ce document.
- Une **solution**, indiquant une démarche possible à adopter pour résoudre le défi
- Une **mise en regard avec les AC**, afin d’expliquer comment le défi s’inscrit dans le cadre du programme national du BUT informatique
- Des **cas limite d’utilisation**, pour déterminer le cadre de réalisation du défi et les limites de celui-ci
- Un **POC**, qui permet de démontrer la viabilité et la réalisation du défi.

Défi 1 : Les mots croisés mais en mieux

Objectif du défi

On définit un tableau à deux dimensions tel que :

```
grid = [
    ['o', 'a', 'n', 'n', 'z', 'x'],
    ['e', 't', 'a', 'e', 'r', 'i'],
    ['i', 'h', 'k', 'e', 'g', 'n'],
    ['i', 'f', 'l', 't', 'o', 'g']
]
```

ou bien :

O	**A**	**N**	**N**	**Z**	**X**
E	**T**	**A**	**E**	**R**	**I**
I	**H**	**K**	**E**	**G**	**N**
I	**F**	**L**	**T**	**O**	**G**

Ainsi qu'une liste de mots :

```
words = ['oath', 'pea', 'eat', 'rain', 'gringe', 'zen', 'gringot']
```

On cherche ici à trouver les mots de la liste présents dans la grille. On peut chercher les lettres pour reconstituer les mots en se déplaçant de manière verticale ou horizontale uniquement. Il est interdit de se déplacer en diagonale ou de sauter les cellules adjacentes. Il est également interdit de repasser sur une lettre pour le même mot.

Résultat attendu : la liste des mots de `words` présents dans `grid`.

Contraintes :

Nous souhaitons que le **test fourni en annexe** dans le POC soit implémenté. Il faudra également imaginer et ajouter un test masqué pour empêcher l'utilisateur de compléter le défi de façon non exhaustive.

Difficulté

Estimée à 5 sur une échelle de 1 à 5, 5 étant le plus difficile. La solution nécessite d'écrire un algorithme relativement complexe pour bien mettre en place un parcours dans la grille. Les diverses conditions (parcours horizontal et vertical uniquement, pas de retour en arrière…) ajoutent une difficulté supplémentaire. Aucun prérequis n'est nécessaire, si ce n'est les concepts algorithmiques de base (boucles, alternatives, assignation de valeurs…).

Mise en regard avec les AC

L'une des approches algorithmiques pour solutionner le problème peut consister en un algorithme DFS de parcours de graphe non orienté où l'on évite les sommets déjà visités.

Le DFS est abordé dans la ressource R2.07 – Graphes, qui cible l'apprentissage critique AC12.01 – Analyser un problème avec méthode (découpage en éléments algorithmiques simples, structure de données...). Cette compétence implique la capacité à décomposer des problèmes complexes en éléments algorithmiques simples, à choisir des structures de données appropriées, et à adopter une approche systématique pour résoudre les problèmes informatiques. Cette compétence vise à développer la capacité des étudiants à résoudre efficacement des problèmes informatiques en utilisant une méthodologie réfléchie.

La résolution d'une telle énigme implique une réflexion algorithmique et une compréhension des structures de données.

Cas limites d'utilisation

Ce test est en réalité plutôt connu, au point qu'il soit utilisé par certaines entreprises pour détecter les bons développeurs à l'aise en algorithmie. Des solutions peuvent facilement être trouvées sur Internet.

Par ailleurs, les solutions sont facilement devinables à la main, en lisant simplement la grille. Des protections devront être mises en place pour éviter une soumission de solution par force brute.

POC

Un exemple d'implémentation algorithmique pour solutionner le problème se trouve en annexes.

Défi 2 : Des requêtes au service du septième art

Objectif du défi

La base de données utilisée pour ce défi est Sakila. C'est une base de données de démonstration MySQL qui simule une entreprise de location de films, avec des tables telles que `film`, `actor`, `customer`, `rental`, et d'autres, permettant aux utilisateurs de pratiquer des requêtes SQL sur des données représentatives d'un contexte réel. Le téléchargement du script de création et d'insertion peuvent se faire à partir de ce lien : https://dev.mysql.com/doc/index-other.html.

L'objectif du défi est de construire un flag prouvant que l'étudiant a correctement réussi à écrire toutes les requêtes.

Enoncé du défi :

Vous êtes sur le point de démarrer une quête cinématographique passionnante à travers la base de données Sakila. Vous allez devoir d'utiliser des requêtes MySQL afin résoudre une série d'énigmes. Votre objectif est de découvrir un flag composé de quatre morceaux : {film-prénom-montant-nom}, par exemple {COAST RAINBOW-LISA-123.15-COOPER). Chaque étape est indépendante et vous rapprochera de ce flag mystérieux.

Étape 1 : le film idéal : Commencez votre aventure en cherchant le film idéal, quelque chose de beau et peut-être même de la science-fiction. Il faut trouver un film avec le mot "beautiful" dans sa description, appartenant à la catégorie "Sci-Fi". Après avoir identifié quelques films correspondant aux critères, plongez dans le monde de la qualité en cherchant celui qui a le plus d'acteurs. Le titre de ce film sera la première partie du flag.

Étape 2 : prénom mystérieux **:** Explorez les prénoms des clients ou des acteurs dont le prénom commence par la lettre 'S'. Choisissez celui qui a le moins de lettres parmi les cinq premiers résultats. Ce prénom constituera la deuxième partie du flag.

Étape 3 : riche client français : Découvrez les clients français et trouvez celui qui a payé le plus cher pour une location. Ce montant sera la troisième partie du flag.

Étape 4 : un cinéphile : Enfin, identifiez le client qui a loué au moins sept films de chaque catégorie, à l'exception de la catégorie 'Documentary'. Trouvez son nom, car cela constituera la dernière partie du flag.

Contraintes

Toutes les requêtes devront être exécutées exclusivement sur la plateforme Déf'IUT, qui mettra à disposition l'environnement MySQL nécessaire. De plus, les résultats de chaque requête doivent être clairement affichés à l'utilisateur. Enfin, aucune modification directe de la base de données Sakila n'est autorisée. Seules des opérations de lecture seront acceptés.

Difficulté

Évalué à 4 sur une échelle de 1 à 5 en termes de difficulté, 5 étant le niveau le plus élevé, ce défi propose un ensemble varié d'étapes nécessitant des compétences avancées en SQL. Chaque étape implique la

formulation de requêtes plus ou moins complexes, depuis l'identification d'un film spécifique selon des critères précis jusqu'à la sélection minutieuse de données à travers des opérations avancées sur la base de données Sakila. La nature progressive des étapes rend ce défi particulièrement exigeant, car les participants doivent démontrer leur maîtrise des concepts SQL à des niveaux croissants de complexité.

Mise en regard avec les AC

Ce défi offre une opportunité aux étudiants de développer et d'appliquer leurs compétences dans le domaine des bases de données et du langage SQL. Il est donc aligné avec les objectifs des ressources R1.05 - Introduction aux bases de données et SQL et R2.06 - Exploitation d'une base de données. En particulier, les participants auront l'occasion de mettre en pratique la compétence AC14.01, qui englobe la capacité à mettre à jour et interroger une base de données relationnelle à travers des requêtes directes ou l'utilisation d'applications dédiées. En résolvant les différentes étapes du défi, les participants peuvent ainsi mettre en lumière leur compétence à interroger de manière efficace une base de données dans un contexte professionnel simulé.

De plus, le défi encourage également le développement de la compétence AC14.02, qui consiste à visualiser des données. Les participants seront amenés à extraire des informations spécifiques de la base de données pour résoudre les énigmes, illustrant ainsi leur aptitude à interpréter visuellement les résultats de leurs requêtes. Ce processus de visualisation des données constitue un élément crucial de la résolution des défis, mettant en avant la pertinence de cette compétence dans le contexte de l'exploitation et de l'analyse de bases de données.

Cas limites d'utilisation

Le défi repose sur la base de données Sakila, et par conséquent, les participants ayant déjà une connaissance approfondie de cette base de données pourraient avoir un avantage, potentiellement réduisant l'équité pour les participants moins familiers avec cette structure spécifique.

POC

En annexe, vous trouverez un exemple de requêtes possibles permettant de résoudre ce défi.

Défi 3 : Point faible : trop fort

Objectif du défi

L'objectif du défi est d'obtenir l'accès au compte d'administration en exploitant une faille de type injection SQL. Le compte administrateur sera identifié à l'aide de l'adresse email : admin@defiut.com.

Contraintes :

Le joueur, dans la description du défi, aura accès à une URL vers un serveur web faisant tourner une application basique. Cette application ne propose que deux pages dont les URL sont les suivantes :

- / : Accueil du site accessible uniquement si l'utilisateur est connecté (redirige vers la page de login s'il ne l'est pas). Cette page contiendra uniquement un texte : le flag permettant de valider le challenge.
- /login : Page de connexion permettant de saisir un email et un mot de passe.

La base de données de type SQLite3 contiendra une seule table nommée *user* et avec trois colonnes :

- id : entier auto incrémenté servant de clé primaire
- email : chaîne de caractère non nulle
- password : chaîne de caractère non nulle (le mot de passe n'est pas nécessairement hashé)

Cette base, au lancement du serveur web, contiendra une seule ligne : le compte administrateur :

- id: 1
- email admin@defiut.com
- password : une chaîne aléatoire de 30 caractères minimum

Lors de la connexion, l'email et le mot de passe entré par l'utilisateur seront concaténés dans la requête SQL afin de la requêter. Il est nécessaire de faire une concaténation pour introduire la faille SQL, si une requête préparée est faite, il ne sera pas possible de valider le défi.

Difficulté

Évalué à 3 sur une échelle de 1 à 5 en termes de difficulté, 5 étant le niveau le plus élevé. La difficulté de ce défi est très relative en fonction de l'utilisateur s'y attaquant. Quelqu'un ayant déjà entièrement conscience du problème saura le détecter et le résoudre très rapidement. Cependant, pour une personne non initiée, il peut être assez difficile de comprendre l'enjeux justifiant le 3. Avec le nombre de ressources sur internet expliquant cette faille, il n'est pas compliqué de vite trouver le chemin à suivre.

Mise en regard avec les AC

Ce défi de sécurité informatique invite à une réflexion centrée sur une base de données. La recherche de la formulation de requêtes s'aligne avec l'objectif de la compétence AC14.01, qui englobe la mise à jour d'une base de données relationnelle ainsi que son interrogation. Étant donné que ce risque de sécurité menace l'intégrité et la confidentialité des données de notre application, l'AC24.02, axée sur la sécurité des données, s'inscrit parfaitement dans les compétences visées par ce défi.

La nécessité de maintenir et développer une solution informatique stable et sécurisée est cruciale dans un environnement en constante expansion. Dans cette perspective, l'acquisition des bonnes pratiques en programmation et conception apparaît comme un aspect essentiel. C'est pourquoi la compétence AC21.03, impliquant cette réflexion, est également intégrée aux objectifs de ce défi. Bien que le risque zéro soit illusoire, il demeure crucial de déployer tous les efforts nécessaires pour garantir la sécurité maximale de notre application.

Ainsi, ce défi requiert une connaissance approfondie des bases de données afin de détecter toute mauvaise pratique en matière de sécurité.

Cas limites d'utilisation

Selon l'implémentation de la requête SQL lors de la connexion, il peut être plus ou moins complexe pour l'utilisateur de valider le défi : si des guillemets sont utilisés plutôt que des apostrophes, etc...

POC

Admettons que la requête pour récupérer l'utilisateur est : SELECT * FROM user WHERE email = "?" AND password = "?" LIMIT 1 où les ? sont à remplacer avec les valeurs à l'aide d'une concaténation.

Il est possible d'injecter : " OR 1=1 -- -- dans le courriel en mettant quelque chose au hasard dans le mot de passe pour ainsi obtenir un accès.

Défi 4 : À corps perdu

Objectif du défi

L'année dernière, vous avez réalisé des programmes répondant à une problématique qui se présente de nouveau. Cependant, vous avez perdu le corps de vos programmes mais vous avez encore vos tests à disposition.

On comptera 3 étapes pour ce défi. À l'échelle de la plateforme, une étape correspond à un seul et unique challenge (à corps perdu (1/3), à corps perdu (2/3) et à corps perdu (3/3)). Ce choix est justifié par la faible difficulté de chaque étape.

Étape 1 : l'utilisateur doit déterminer la durée en minutes (sortie) d'un trajet à partir de deux horaires (entrées en chaine de caractères) de départ et d'arrivée.

Étape 2 : À partir d'une chaine de caractère en entrée, l'utilisateur doit afficher en sortie cette chaine x fois (où x est la longueur de la chaine) en effectuant une permutation de l'ordre des lettres (exemple : café, aféc, féca, écaf).

Étape 3 : À partir d'une chaine de caractère en entrée, l'utilisateur doit effectuer un traitement sur celle-ci afin que l'affichage final soit une combinaison de cette chaine avec la chaine inverse (exemple : Holiday ? → HyoaldiidlaoyH).

Illustration :

L'utilisateur se voit recevoir les entrées et les sorties suivantes :

Entrée	Sortie attendue
8:12 14:32	380

Entrée	Sortie attendue
9:17 23:17	840

Entrée	Sortie attendue
6:54 12:13	319

L'utilisateur va dans un premier temps devoir réfléchir à propos de la logique entre l'entrée et la sortie. Une fois qu'il a compris, il construit un programme qui permet de recréer cette logique. Quand l'utilisateur a une proposition, il lance une batterie de tests (avec les sorties qu'il voit mais aussi des tests cachés, pour éviter les résolutions de type brute force), qui valide le sous-défi si elle passe à 100%.

Le flag pour valider le défi sur la plateforme est une chaine de caractère communiquée à l'utilisateur une fois le code fourni exécuté et validé par le serveur.

Contraintes :

Nous demandons en priorité que les langages utilisés par le public-cible soient à leur disposition afin qu'ils puissent proposer des solutions. Les programmes doivent pouvoir être réalisés après un clic sur un sous-défi.

Nous souhaitons également que les **tests fournis en annexe** soient implémentés et affichés à l'utilisateur afin de valider les défis et le guider vers la compréhension de l'algorithme à réaliser. Il faudra également imaginer et ajouter des tests masqués pour empêcher l'utilisateur de compléter le défi de façon non exhaustive.

Difficulté

Ce défi est classé difficulté 1/5. Le temps consacré à celui-ci dépend plutôt du temps passé à comprendre le lien entre l'entrée et la sortie. Une fois la logique comprise, la réalisation du programme devrait être rapide.

Mise en regard avec les AC

En lien avec les AC 11.02 et 11.03, ce défi fait appel à la capacité de compréhension, d'analyse et de réalisation de programme afin de réaliser des tâches simples. Dans un contexte professionnel, il est possible que les tests soient réalisés par une personne spécialisée, et ce défi place les utilisateurs dans une situation où ils doivent implémenter le programme répondant à ces tests, ces exigences.

Il est également demandé à l'équipe de réalisation de proposer et d'implémenter des tests pertinents afin de valider convenablement la réussite de chaque sous-défi. Cela va permettre de vérifier la qualité des programmes réalisés par les utilisateurs de la plateforme. Ces compétences sont totalement en lien avec l'AC 21.04.

Cas limites d'utilisation

Une manière très simple de résoudre chaque étape serait de procéder par force brute : si entrée alors retourner sortie, sinon si entrée retourner sortie2... Évidemment, on cherche ici à trouver une solution exhaustive. De fait, comme susmentionné, des protections doivent être implémentées.

POC

Un exemple d'implémentation pour chaque étape se trouve en annexes.