ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

MINI PROJET

2024/2025

DSL pour un langage de commande domotique

« Compilation 2 »

Ingénierie Informatique et Réseaux

Réaliser par :

Oualid Elmansour
Saad Elblaidi
Anas Lahouaoui
Mohamed Lhbari
Mouad Houssaddin

Encadrée par :
Pr. Nadiri

1. Introduction

A. Contexte général

Présentez brièvement la domotique et son importance dans la gestion des appareils domestiques intelligents. Expliquez l'intérêt d'un langage spécifique à ce domaine (DSL) pour simplifier l'interaction avec ces systèmes.

B. Objectif du projet

Précisez l'objectif de ce projet, qui est de concevoir un DSL permettant de formuler des commandes domotiques de manière claire et intuitive.

C. Problématique

Pourquoi créer un DSL pour la domotique et en quoi cela facilite-t-il l'interaction avec les appareils ? Soulignez la nécessité d'une compilation efficace pour interpréter ces commandes.

2. Théorie de la Compilation

La compilation est le processus par lequel un programme écrit dans un langage de programmation est transformé en un autre format, souvent un code machine ou un code intermédiaire. Cette transformation est effectuée par un **compilateur**, qui est un programme spécialisé dans ce travail.

La compilation se divise généralement en plusieurs étapes, chacune ayant un rôle spécifique dans la transformation du programme source (le code écrit par le programmeur) en un programme exécutable. Ces étapes comprennent l'analyse lexicale, l'analyse syntaxique, l'analyse sémantique, la génération de code, et l'optimisation.

A . Analyse Lexicale (Lexing)

L'analyse lexicale est la première étape du processus de compilation. Elle consiste à transformer le code source (texte brut) en une série de symboles appelés tokens. Chaque token représente une unité lexicale, telle qu'un mot-clé, un identifiant, un opérateur ou une constante.

Par exemple, dans une commande de votre DSL :

```
Allumer lampe salon à 20:30
```

Les tokens pourraient être :

- `Allumer` : Action
- `lampe` : Appareil
- `salon` : Lieu

L'analyse lexicale est effectuée par un **analyseur lexical** (ou **lexer**), qui prend le code source comme entrée et le transforme en une liste de tokens.

B. Analyse Syntaxique (Parsing)

L'analyse syntaxique est l'étape où les tokens générés par l'analyseur lexical sont regroupés pour former une **structure syntaxique** sous forme d'un **arbre syntaxique** ou d'un **graphe de dérivation**. Cette étape vérifie si le code source respecte les règles de grammaire du langage.

Par exemple, dans votre DSL, la grammaire peut être définie comme suit :

```
Commande -> Action Cible Article Horraire
Action -> allumer | éteindre | régler
Cible -> Appareil Lieu
Horraire -> 'à' H ':' M | epsilon
```

L'analyseur syntaxique (ou **parser**) s'assure que les tokens reçus suivent cette structure. Si la commande est mal formée, par exemple si l'action est manquante ou mal placée, l'analyseur générera une erreur de syntaxe.

C. Analyse Sémantique

L’analyse sémantique se concentre sur le sens du programme. Elle vérifie que les opérations et les instructions sont logiquement valides dans le contexte du langage. Par exemple :

- Vérifier qu'une action est possible pour un appareil donné (par exemple, allumer une lampe).
- S'assurer qu'une horaire est bien dans une plage valide (par exemple, une heure entre 00:00 et 23:59).

Dans un langage de commande domotique, cela peut aussi inclure la vérification des cibles et des actions disponibles pour des appareils spécifiques. L'analyse sémantique peut impliquer la construction de tables de symboles ou la vérification de la compatibilité des types (heure, appareil, etc.).

D. Génération de Code

La génération de code est l'étape où l'arbre syntaxique (ou la représentation intermédiaire) est transformé en une forme exécutable. Pour un DSL de commandes domotiques, cette génération de code pourrait être la traduction des commandes en instructions compréhensibles par un serveur domotique, un microcontrôleur, ou tout autre système capable d'exécuter ces actions.

Par exemple, une commande comme "allumer lampe salon" pourrait être traduite en un appel à une API, une commande HTTP ou un message MQTT envoyé à un appareil.

La génération de code peut se faire sous plusieurs formes :

- **Code machine** : Transformation en code binaire compréhensible par le processeur.
- **Code intermédiaire** : Utilisation de langages comme bytecode (par exemple, Java).
- **Représentation fonctionnelle** : Transformation en appels API ou en commandes spécifiques à un système domotique.

E. Optimisation

L'optimisation est une étape qui intervient après la génération de code pour améliorer les performances du code généré. Elle consiste à rendre le code plus rapide, plus compact ou plus efficace sans changer son comportement. Dans le cas d'un DSL de commande domotique, l'optimisation pourrait viser à réduire le temps de réponse ou à minimiser l'usage des ressources du système.

Les optimisations peuvent concerner :

- **Optimisation du temps d'exécution** : Réduction des instructions superflues.
- **Optimisation de la mémoire** : Réduction de la taille des messages envoyés ou du stockage des configurations.

3. Description du DSL de Commandes Domotiques

Le DSL (Domain-Specific Language) développé pour ce projet est conçu pour fournir un moyen simple et intuitif de contrôler des appareils domotiques au sein d'une maison ou d'un bâtiment. Il permet aux utilisateurs de formuler des commandes en langage naturel, organisées autour des éléments fondamentaux suivants : **Action**, **Cible**, **Lieu**, et une **Horaire optionnelle**.

A. Structure Générale du DSL

Chaque commande suit une structure bien définie et facile à comprendre :

```
Commande -> Action Cible Lieu Horraire
Action -> Ensemble des actions disponibles
Cible -> Ensemble des appareils disponibles
Lieu -> Ensemble des lieux disponibles
Horraire -> 'à' H ':' M | epsilon
H -> heures (de 00 à 23)
M -> minutes (de 00 à 59)
```

B. Composants du DSL

- **Action**
    - L'Action décrit ce que l'utilisateur souhaite effectuer. Elle représente le **verbe** de la commande.
    - Exemples d'actions disponibles :
        - **allumer** : pour activer un appareil (ex : allumer une lampe).
        - **éteindre** : pour désactiver un appareil (ex : éteindre un chauffage).
        - **régler** : pour configurer un paramètre spécifique d'un appareil (ex : régler une température).

- **Cible**
    - La Cible désigne l'appareil concerné par l'action.
    - Exemples de cibles possibles :
        - **lampe**
        - **chauffage**
        - **thermostat**
        - **porte**

- **Lieu**
    - Le Lieu permet de préciser où l'action doit être effectuée.
    - Exemples de lieux possibles :
        - **salon**
        - **chambre**
        - **cuisine**
        - **bureau**

- **Horraire (optionnelle)**

    - L'Horaire permet d'indiquer quand l'action doit être exécutée. Si elle est absente, l'action est effectuée immédiatement.
    - Le format de l'horaire est :

      ```
      'à' H ':' M
      ```

    - Exemples :
        - `"à 18:30"` : Exécute l'action à 18h30.
        - **epsilon (vide)** : Exécute l'action immédiatement.

C. Exemple de Commandes

Voici quelques exemples illustrant l'utilisation du DSL :

- **Commande immédiate sans horaire** :

    - `"allumer lampe salon"` : Allume immédiatement la lampe dans le salon.
    - `"éteindre chauffage chambre"` : Éteint immédiatement le chauffage dans la chambre.

- **Commande avec horaire** :

    - `"allumer lampe salon à 20:00"` : Allume la lampe du salon à 20h00.
    - `"régler thermostat cuisine à 18:30"` : Régler le thermostat de la cuisine à 18h30.

D. Grammaire Formelle

La grammaire formelle de ce DSL est définie comme suit :

```
Commande -> Action Cible Lieu Horraire
Action -> allumer | éteindre | régler
Cible -> "lampe" | "chauffage" | "thermostat" | "porte"
Lieu -> "salon" | "chambre" | "cuisine" | "bureau"
Horraire -> 'à' H ':' M | epsilon
H -> heures (00 à 23)
M -> minutes (00 à 59)
```

E. Justification du DSL

1. **Simplicité** : Le langage est conçu pour être proche du langage naturel, facilitant son adoption par les utilisateurs non techniques.
2. **Clarté** : Les éléments de la commande (Action, Cible, Lieu, Horraire) sont séparés de manière explicite, ce qui réduit les risques d'ambiguïté.
3. **Flexibilité** : L'horaire est optionnelle, permettant à l'utilisateur de choisir entre des actions immédiates ou planifiées.
4. **Extensibilité** : Le DSL peut facilement être enrichi avec de nouvelles actions, cibles, ou lieux à mesure que le système évolue.

F. Conclusion

Ce DSL offre une solution intuitive et efficace pour contrôler les appareils domotiques en utilisant des commandes textuelles. Sa simplicité et sa structure claire permettent une utilisation fluide, même pour des utilisateurs non techniques, tout en étant suffisamment flexible pour s'adapter aux besoins futurs.

4. Conception de l'Analyseur Lexical et Syntaxique

A. Analyse Lexicale

L'analyseur lexical décompose une commande en éléments simples appelés **tokens**. Ces tokens sont les composants de base, tels que les actions, cibles, lieux ou horaires.

- **Objectif** : Identifier les mots-clés et les éléments essentiels de la commande.
- **Exemple de Tokens** :
  - `"allumer"` → **ACTION**
  - `"lampe"` → **CIBLE**
  - `"salon"` → **LIEU**
  - `"à 20:30"` → **HORRAIRE**
- **Commande d'exemple** :

```
allumer lampe salon à 20:30
```

- **Tokens générés** :

```
 [ACTION: allumer], [CIBLE: lampe], [LIEU: salon], [HORRAIRE: à
20:30]
```

B. Analyse Syntaxique

L'analyseur syntaxique vérifie que la séquence de tokens respecte les règles définies dans la grammaire du DSL. Il construit également une structure organisée de la commande.

- **Objectif** : S'assurer que la commande est valide et complète.
- **Grammaire utilisée** :

```
Commande -> Action Cible Lieu Horraire
Action -> allumer | éteindre | régler
Cible -> lampe | chauffage | thermostat | porte
Lieu -> salon | chambre | cuisine | bureau
Horraire -> 'à' H ':' M | epsilon
```

- **Exemple d'Analyse Syntaxique** :

```
allumer lampe salon à 20:30
```

- La structure générée est :

```
Commande
├── Action: allumer
├── Cible: lampe
├── Lieu: salon
└── Horraire: à 20:30
```

C. Conclusion

L'analyse lexicale et syntaxique garantit que chaque commande respecte les règles du DSL. Cette approche permet de valider les commandes, d'assurer leur exécution correcte et de faciliter la détection des erreurs.

5. Réalisation

Dans cette section, nous décrivons la mise en œuvre technique du projet, en détaillant les étapes clés, les outils utilisés et les résultats obtenus. Nous incluons également des captures d'écran pour illustrer chaque étape importante.

A. Environnement de Développement

- **Langage utilisé** : Java
- **IDE** : Eclipse
- **Outils utilisés** :
    - **JavaCC** pour l'analyse lexicale et syntaxique.
    - Bibliothèques standard de Java pour la gestion des commandes et de la logique.

B. Interface Utilisateur

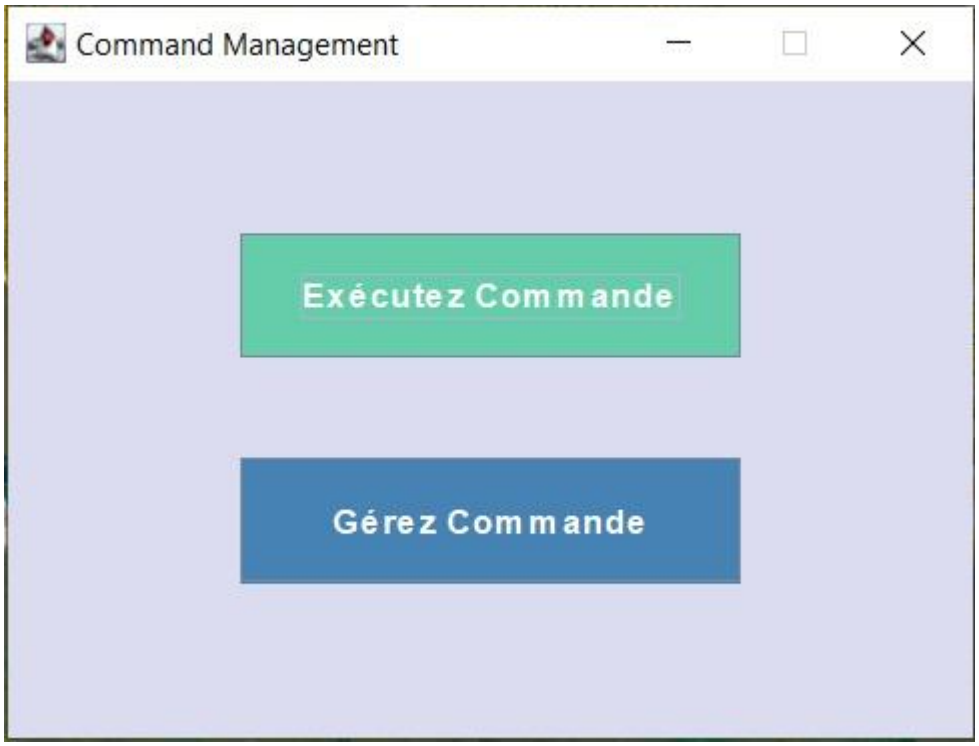

Figure 1 : Main Menu

Figure 2 : Gestion des commandes

Figure 3 : Les choix possibles

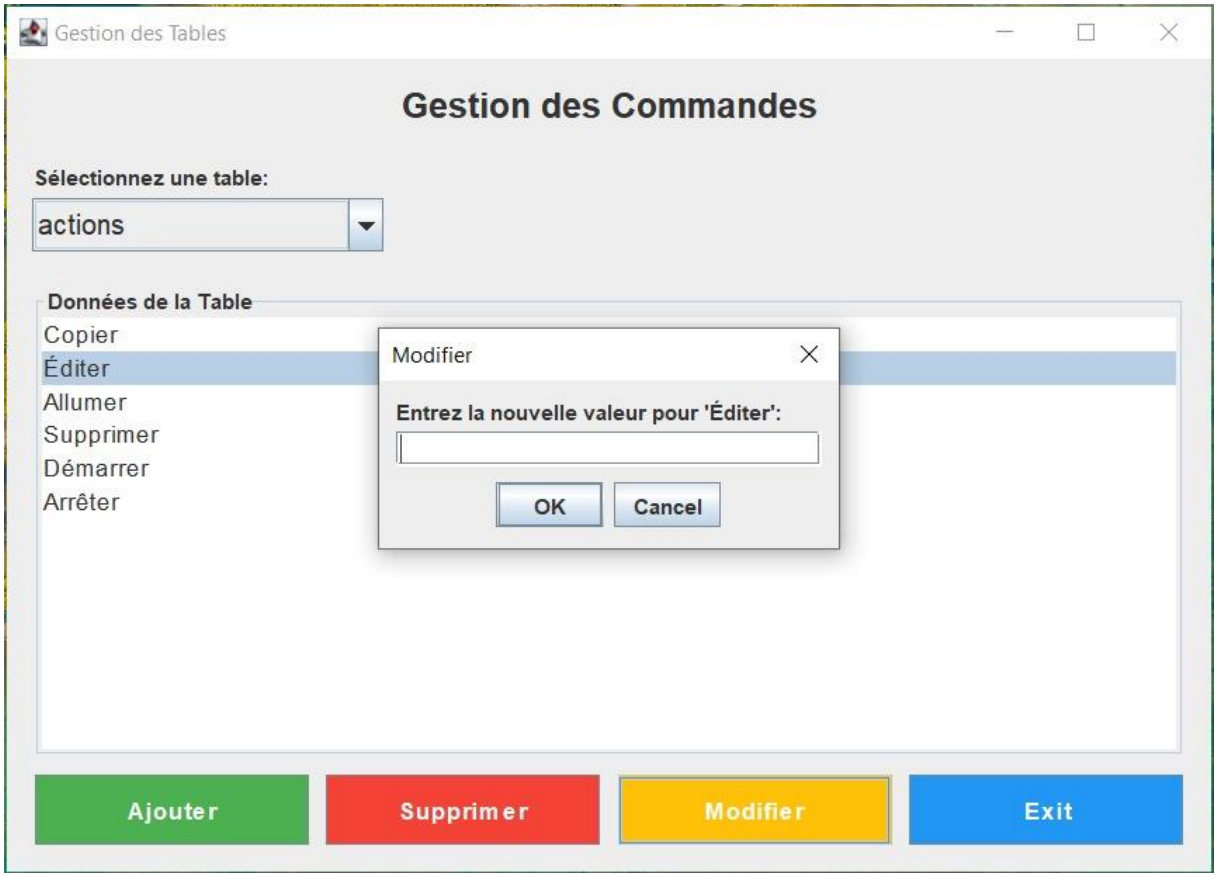

Figure 4 : Pour Modifier une valeur

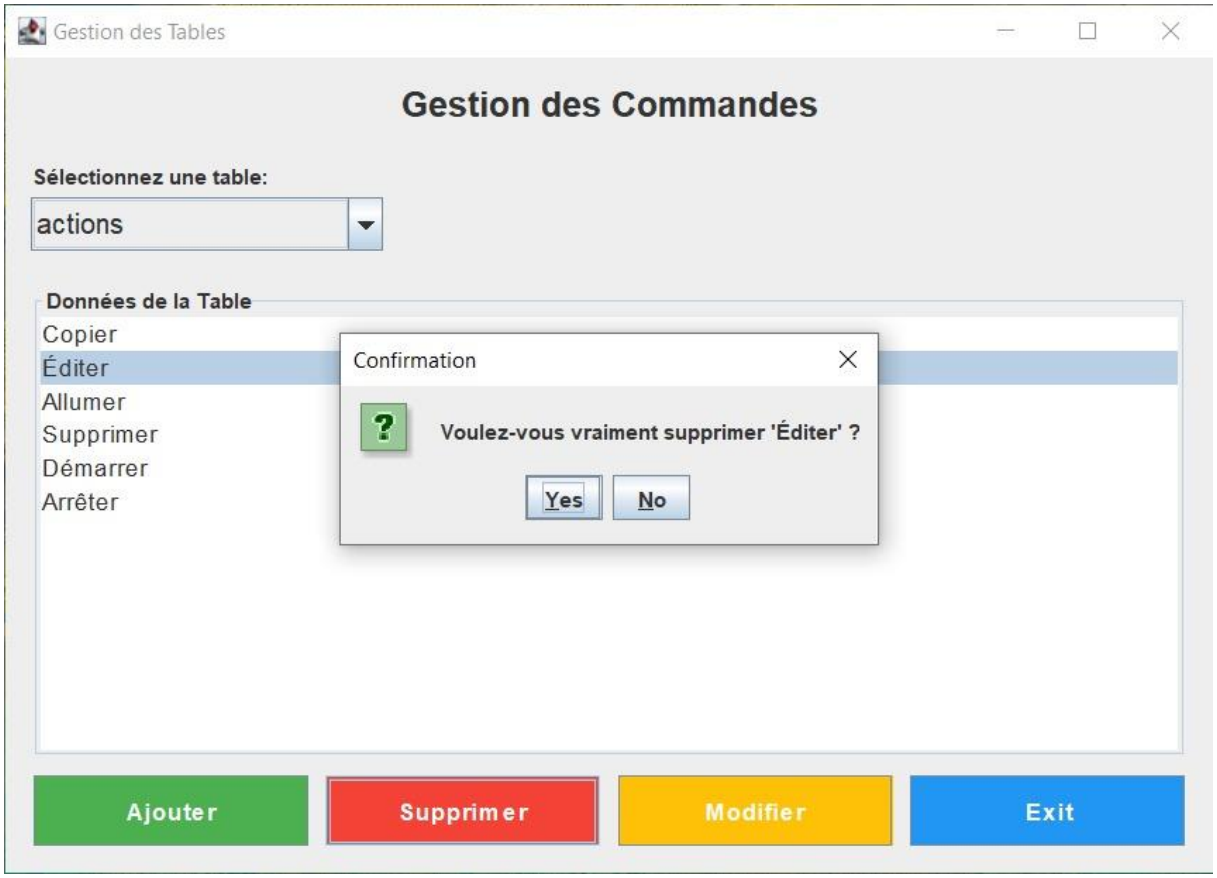

Figure 5: Supprimée une valeur

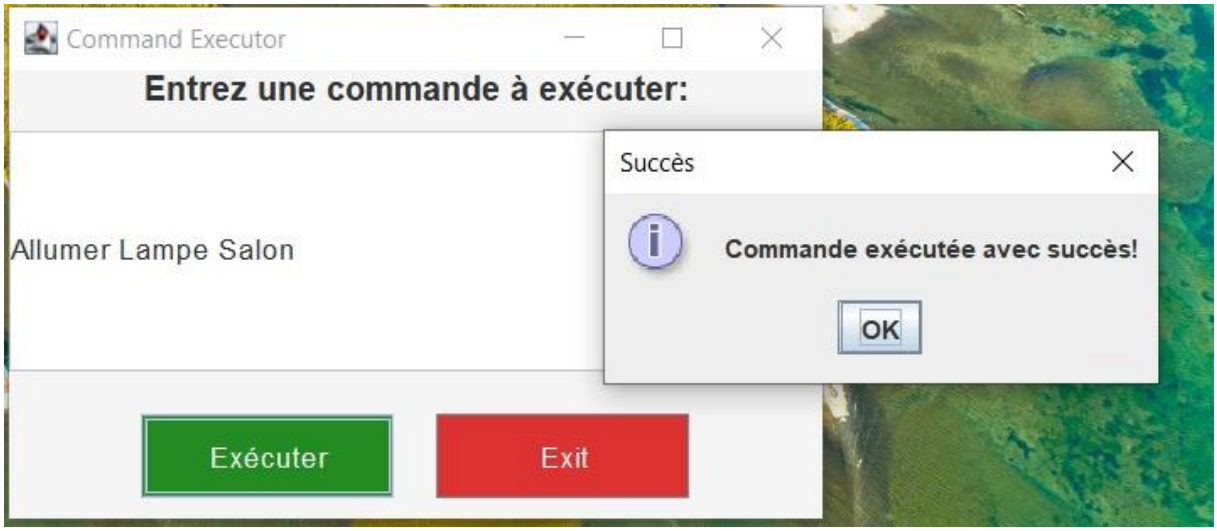

Figure 6 : Test 1

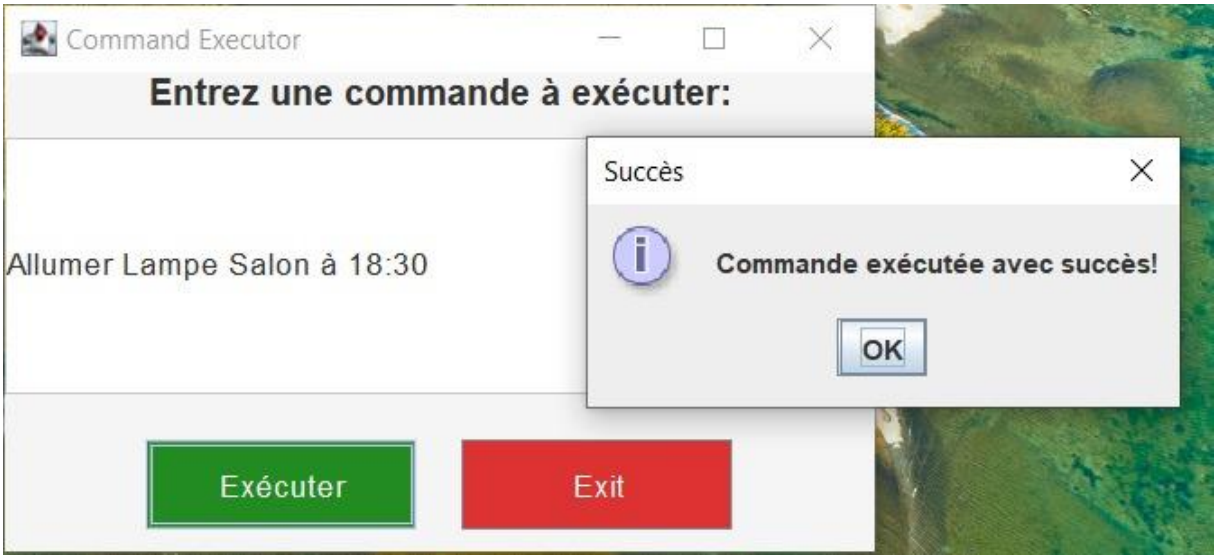

Figure 7: Test 2

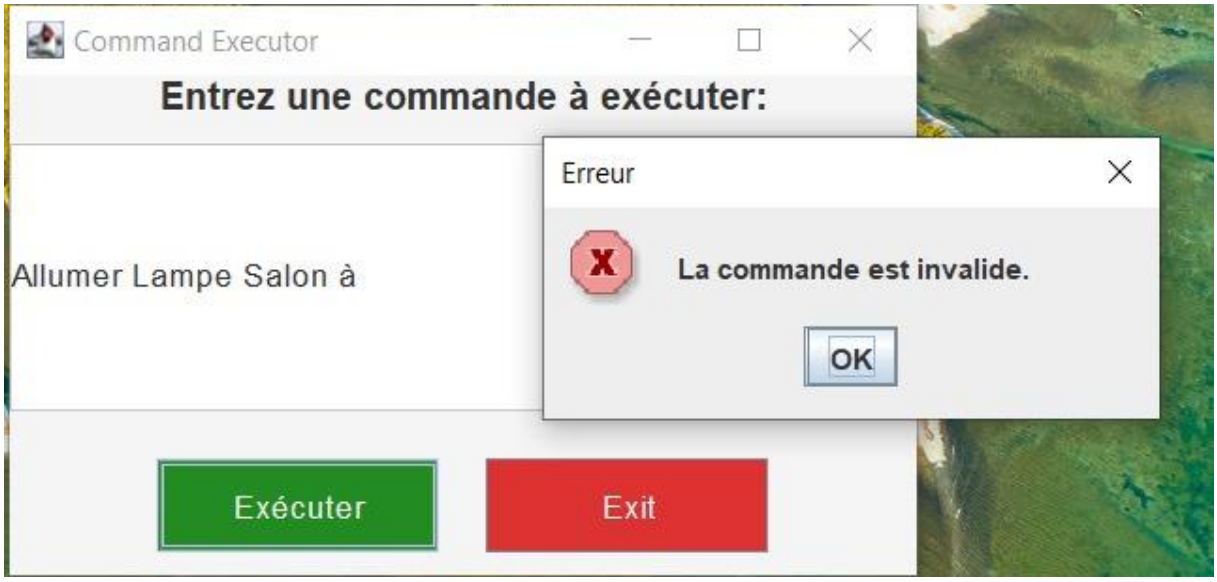

Figure 8 : Test 3

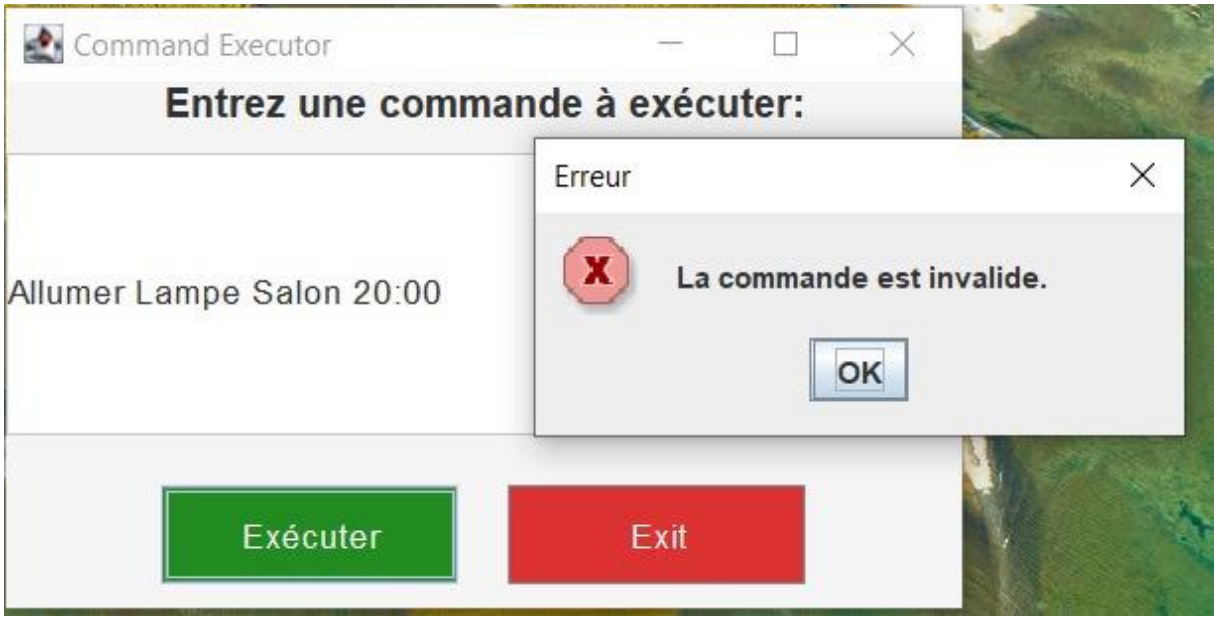

Figure 9 : Test 4

6. Conclusion et Perspectives

A. Conclusion

Le projet de développement d'un DSL (Domain-Specific Language) pour les commandes domotiques a permis de répondre aux besoins spécifiques de ce domaine en offrant une syntaxe claire et intuitive pour contrôler divers appareils dans un environnement domestique.

Les étapes clés réalisées incluent :

- La définition d'une grammaire adaptée à la gestion des actions, cibles, lieux et horaires.
- L'implémentation d'un analyseur lexical et syntaxique permettant de valider et d'interpréter les commandes.
- La gestion des erreurs avec des messages explicites pour guider l'utilisateur en cas de commandes incorrectes.

Ce projet a également permis d'approfondir des concepts fondamentaux en compilation, tels que l'analyse lexicale, l'analyse syntaxique et la gestion des grammaires. Grâce à ces bases solides, le DSL développé est fonctionnel et extensible pour des besoins futurs.

B. Perspectives

Plusieurs améliorations et extensions peuvent être envisagées pour enrichir ce projet :

Extensions Fonctionnelles

- Ajout de nouvelles actions (par exemple, *augmenter*, *réduire*, etc.).
- Support pour des commandes plus complexes, comme des séquences d'actions ou des conditions (ex. : *"si la température dépasse 25°C, éteindre le chauffage"*).

Interface Graphique

- Développement d'une interface utilisateur conviviale permettant de saisir et d'exécuter les commandes via une application graphique.

Support Multilingue

- Adaptation du DSL pour accepter des commandes dans différentes langues (français, anglais, etc.).

Intégration avec des systèmes réels

- Connexion du DSL à des systèmes domotiques existants (ex. : Home Assistant, MQTT) pour exécuter les commandes directement sur des appareils physiques.

Optimisation des Performances

- Amélioration de l'efficacité des analyseurs pour traiter des commandes longues ou complexes.

Validation Avancée

- Implémentation d'un analyseur sémantique pour vérifier la cohérence des commandes (par exemple, empêcher de *"chauffer la climatisation"*).