REPUBLIQUE DU CAMEROUN
****
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
****
UNIVERSITE INTER-ETATS CONGO-CAMEROUN
****
ECOLE SUPERIEURE INTERNATIONALE DE GENIE NUMERIQUE
****

REPUBLIC OF CAMEROON
****
MINISTRY OF HIGHER EDUCATION
****
CONGO-CAMEROUN INTER-STATES UNIVERSITY
****
INTERNATIONAL ADVANCED SCHOOL OF DIGITAL ENGINEERING
****

DEVOIR DE THEORIE LANGAGE ET COMPILATION

MEMBRES DU GROUPES

1-NKOTTO FRANCK DYLAN
2-NKOTO BRANDON PARFAIT JUNIOR
3-EYINGA ONDOA PELZA
4-BIDIAS AMBASSA MIKE
5-AKONO JORDAN
6-DOBGIMA DARIL
7-FEUWO BIBIANG EPHREIM (chef)
8-ANGONI IVON ROSSI
9-ALIMA LOUIS
10-NALINGUI JUDICAEL ARIANE VICTOIRE
11-ONANA MBA MICHEL
12-MEZANGA M'ENGOLO
13-ZO'O NYABA ASSE JORDAN
14-ANGO ABONDO
15-MBEGA MVOGO

Enseignant : Etienne KOUOKAM

Introduction

Dans le but d'assoir les notions vues dans le cours de théorie de langage et compilation en troisième année du cycle d'ingénierie des systèmes numériques, il nous a été demandé de produit un logiciel pour tester un automate défini. . Entendez par là, un programme permettant à un utilisateur donné, grâce à une interface conviviale mis à sa disposition, d'introduire des informations/données liées à un automate fini quelconque (AFD, AFDC, Ɛ-AFN) qu'il a en main. Une fois ces informations/données introduites, le simulateur doit pouvoir simuler ledit automate sur une chaîne en entrée introduite par l'utilisateur, qui pourra répéter le processus autant de fois qu'il le souhaite, éventuellement sur des chaînes déférentes. Le message affiché par votre programme dira si le mot introduit réussit. Le cas échéant, il devra indiquer à l'utilisateur en quoi la chaîne en entrée n'est pas acceptée et doit pouvoir gérer les effets de bord. C'est dire la place qu'occupe la gestion des erreurs dans votre simulateur.

REPARTION DES TACHES PAR MEMBRES

MEMBRES	**TACHES**
MBEGA MVOGO	Test AFD
ANGO ABONDO	Réalisation des interfaces
ZO'O NYABA ASSE JORDAN	Test AFD
MEZANGA M'ENGOLO	Conception de l'algorithme de l'AFN et saisie du document Word
ONANA MBA MICHEL	Test AFD
NALINGUI JUDICAEL ARIANE VICTOIRE	Conception des interfaces
ALIMA LOUIS	Recherche sur internet
ANGONI IVON ROSSI	Programmation AFN
FEUWO BIBIANG EPHREIM (chef)	Programmation de l'AFD et assemblage du code
DOBGIMA DARIL	Conception de l'algorithme du AFD et conception interfaces
AKONO JORDAN	test AFD et montage du document Word
BIDIAS AMBASSA MIKE	Assemblage du code
EYENGA ONDOA PELZA	Test Ɛ -AFN
NKOTO BRANDON PARFAIT JUNIOR	Test AFN
NKOTTO FRANCK DYLAN	

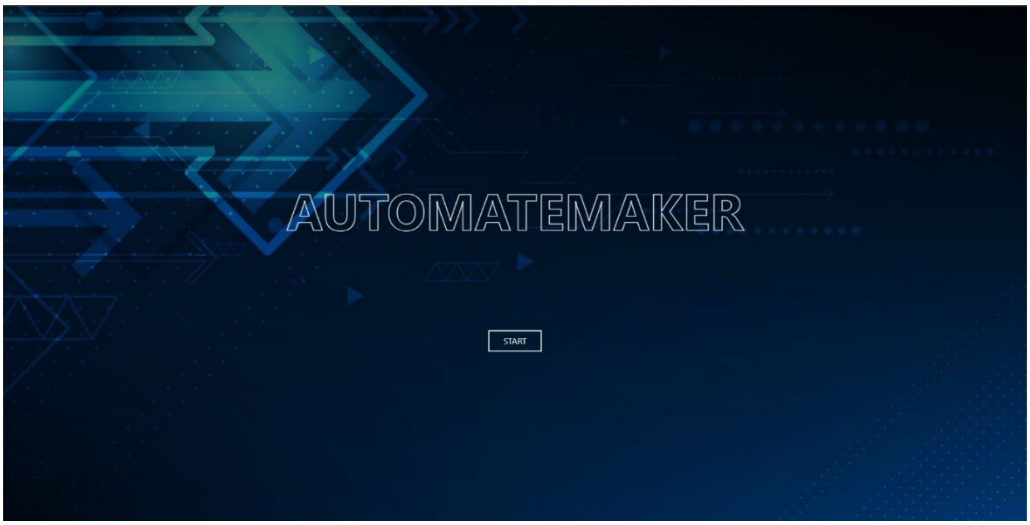
Choix du langage

JavaScript : pour la logique de l'application et implémenter l'algorithme des automates

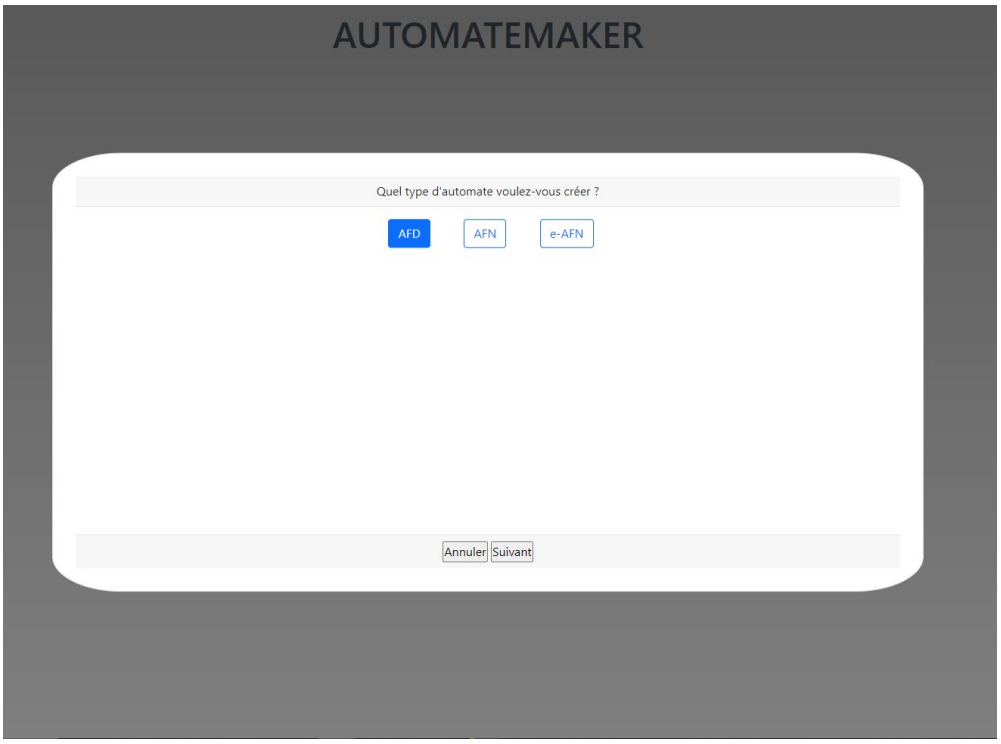
HTML & CSS : pour l'interface graphique

Description de l'application

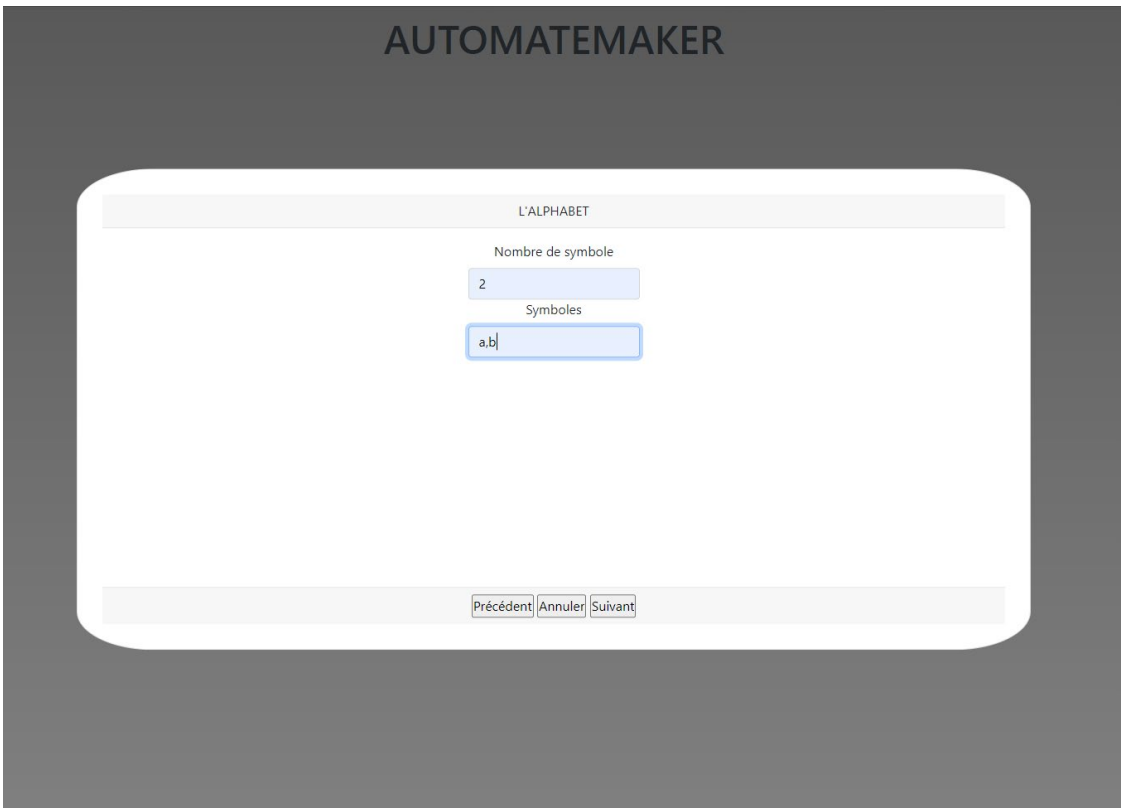
Sur la première interface de l'application, on a la possibilité de lancer l'application



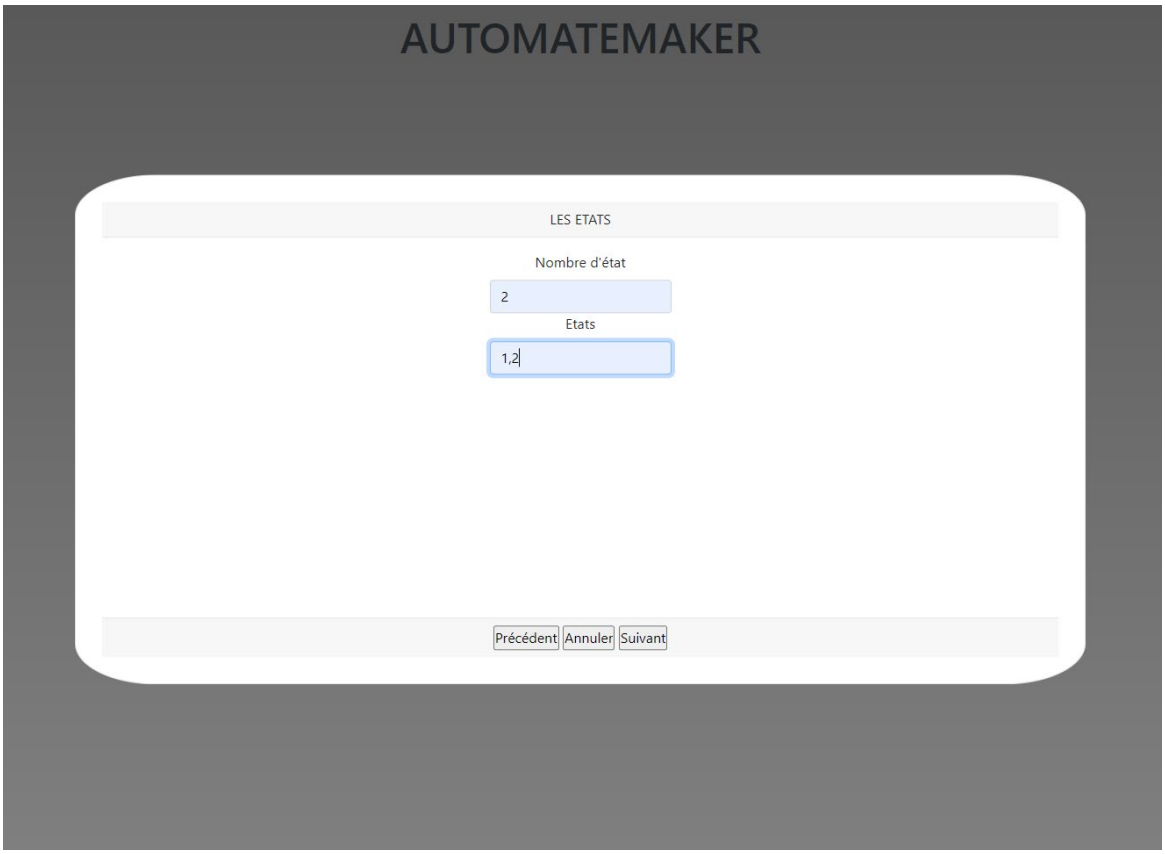
Après le lancement de l'application, on a la possibilité de choisir un type d'automate (AFD, AFN, e-AFN). Malheureusement, on a pu implémenter l'AFD et L'AFN



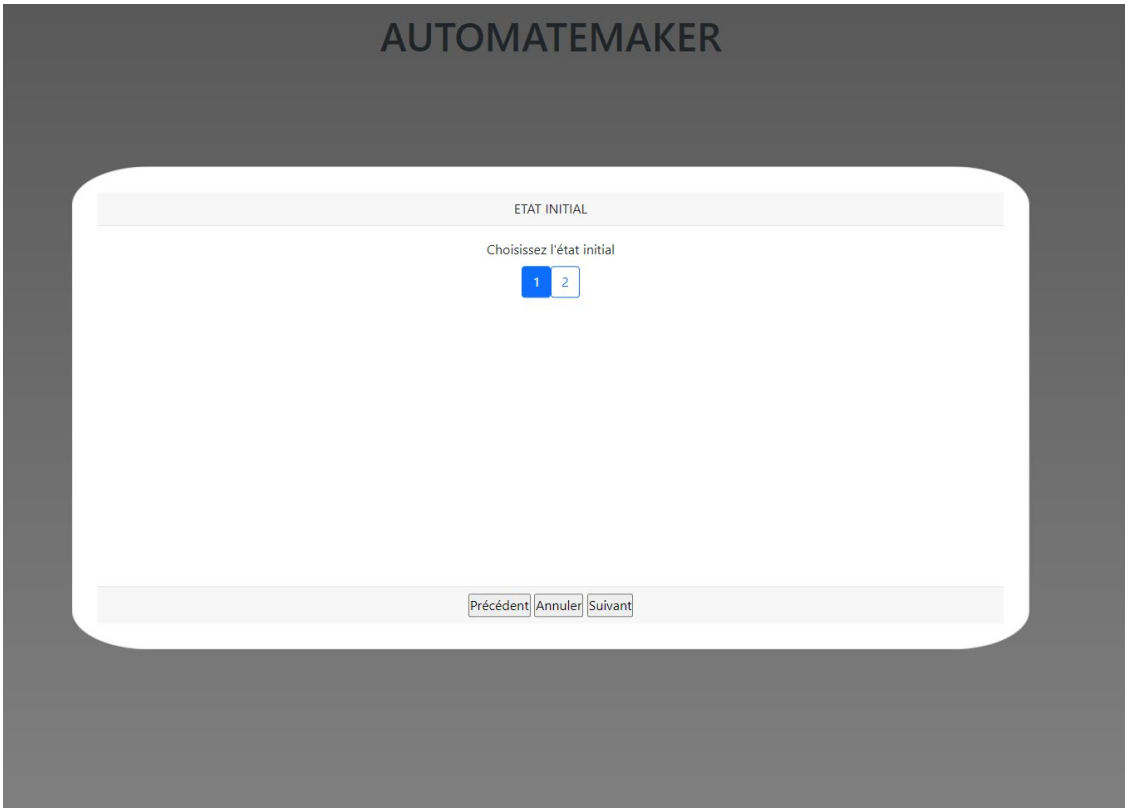
Puis on définit le nombre de symbole et les symboles de l'automate

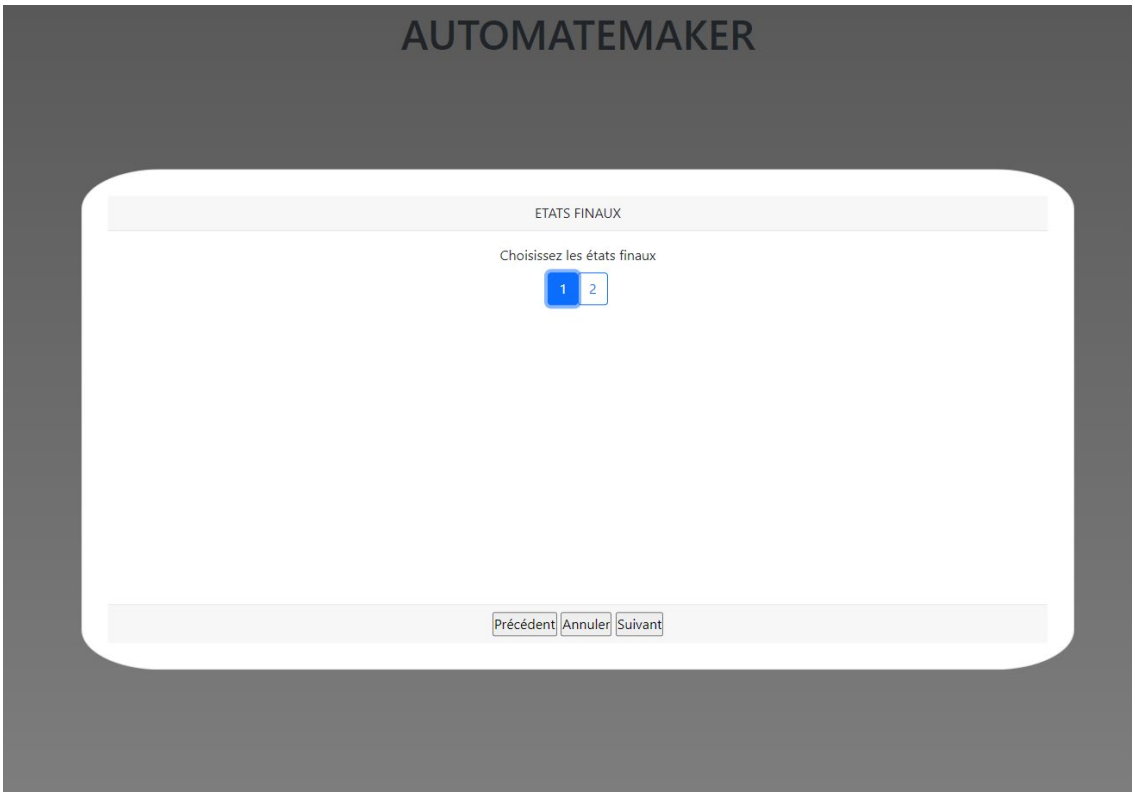


Ensuite on définit le nombre d'états et les symboles qui représentent les états

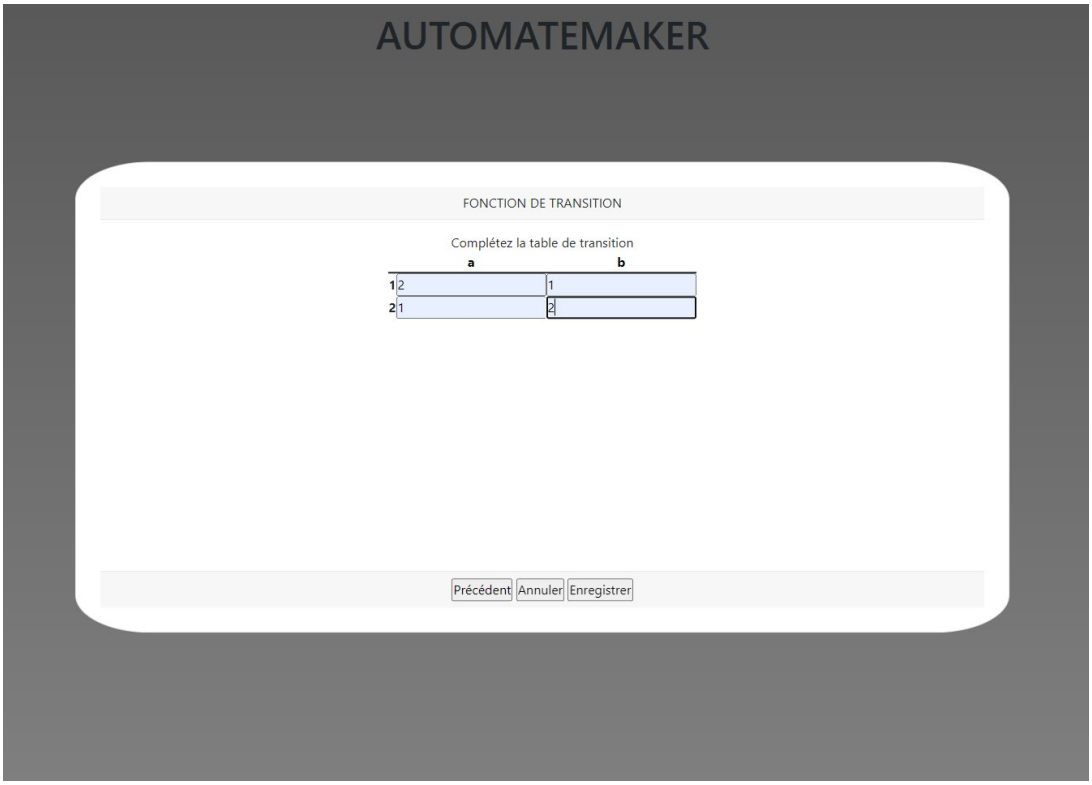


Apres on définit parmi les états, ceux qui sont initiaux et ceux qui sont finaux

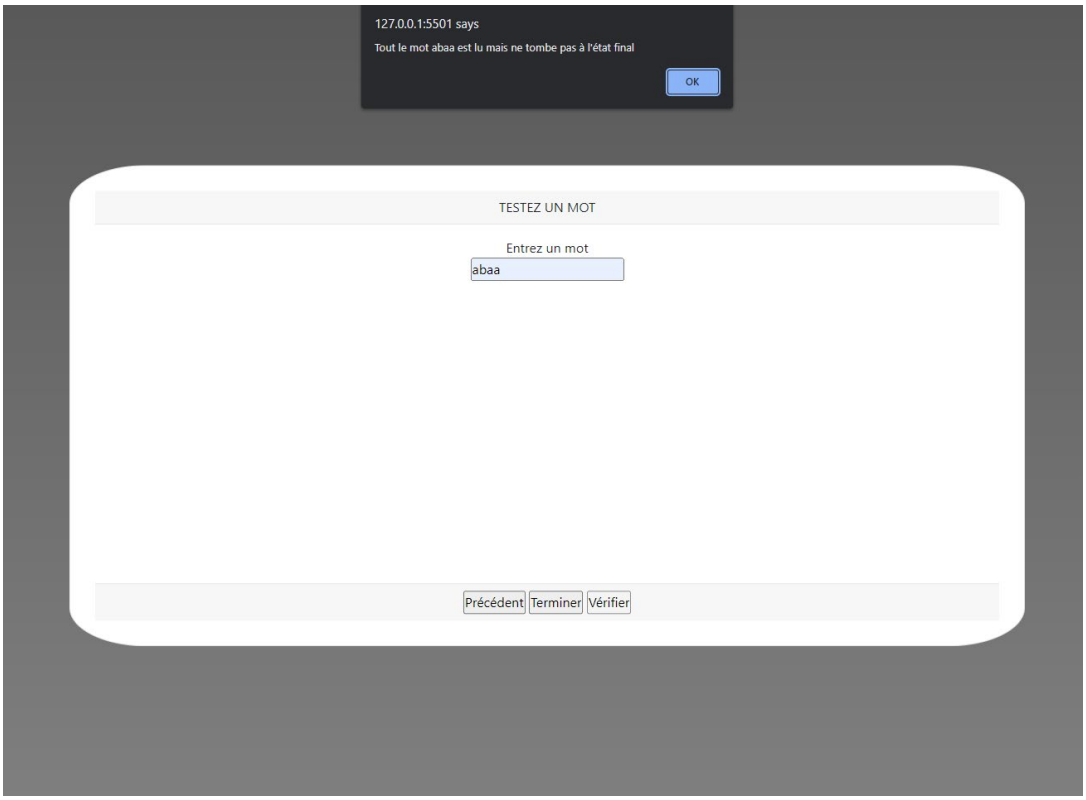


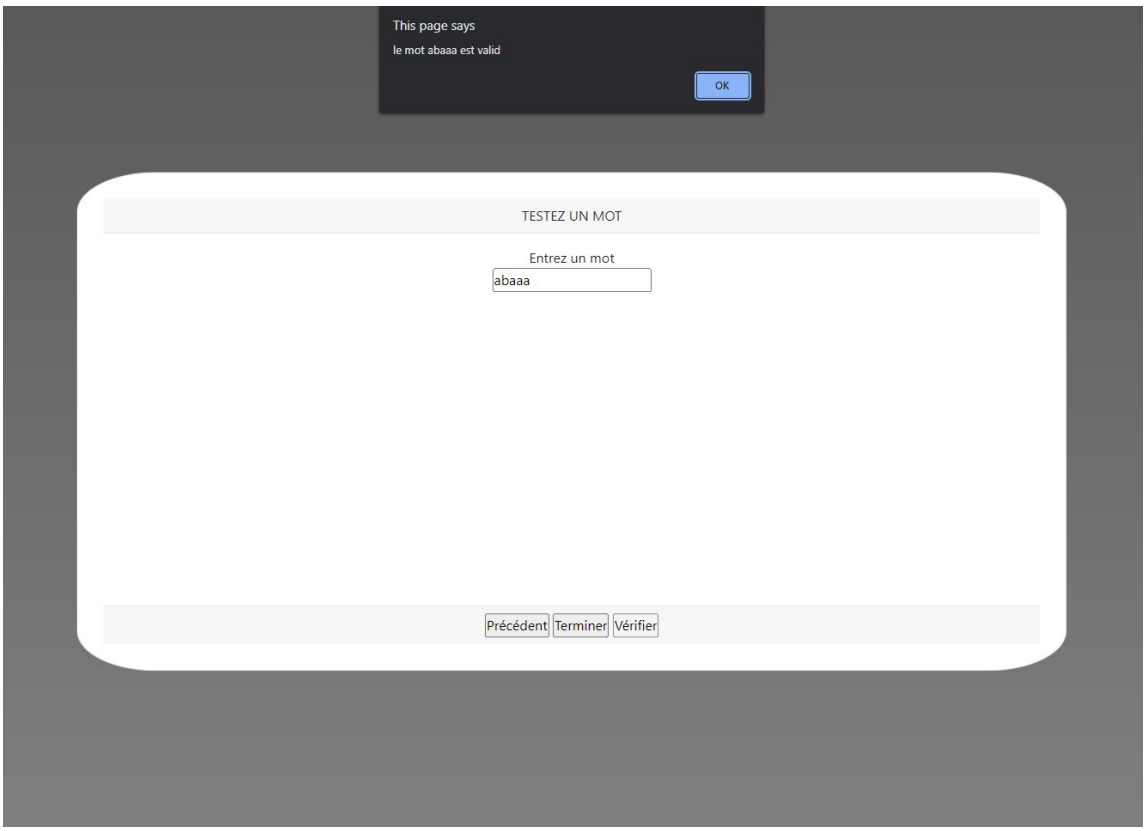


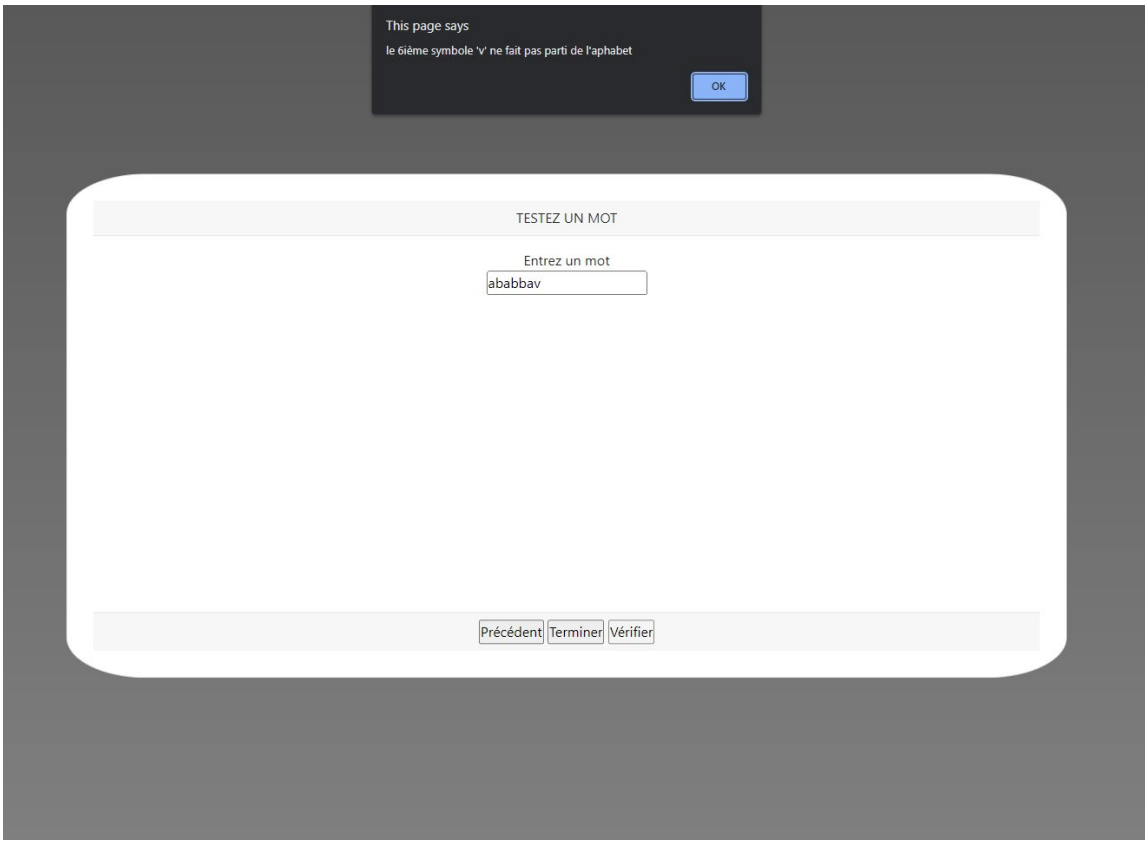
Puis, on remplit la table de transition qui a été construite grâce à l'indication de nombre d'états et du nombre de symboles.



Ensuite on peut tester si un mot est reconnu par l'automate qu'on a défini


127.0.0.1:5501 says
Tout le mot abaa est lu mais ne tombe pas à l'état final
OK
TESTEZ UN MOT
Entrez un mot
abaa
Précédent
Terminer
Vérifier


This page says
le mot abaaa est valid
OK
TESTEZ UN MOT
Entrez un mot
abaaa
Précédent
Terminer
Vérifier



Algorithme (AFN)

Pour reconnaitre un mot, on parcourt l'automate, lorsqu'on arrive sur une indétermination (plusieurs chemins possibles) on parcourt un des chemins puis à la fin du processus, on vérifie si l'état courant fait partie des états finaux dans le cas contraire on essaye le deuxième chemin en faisant une procédure récursive.

create.html | script.js | style.css 1

Théorie des langages > Projet > js > script.js > testresult

```
function testWordAFN(mot, init) {
    let word = mot.toString().split('');
    let current = init;
    let i = 0;
    console.log("init:-->" + current);
    while (!automate.validAFN && i < word.length) {
        if (/,/.test(current)) {
            currents = current.split(',');
            wordR = mot.substr(i, mot.length);
            currents.forEach(current => {
                console.log(wordR);
                console.log("brache : ")
                if (word.length == 0) {
                    if (automate.etatsFinaux.indexOf(current) != -1) {
                        automate.validAFN = true;
                        console.log("brache: valid");
                    }
                } else {
                    testWordAFN(wordR, current);
                }
            });
        } else {
            if (current == "") {
                //console.log("il n'y a pas transition sur " + prevCurrent + " avec le symbole " + character);
            } else {
                console.log("init : " + current + " word :  " + word[i])
                current = automate.tableTransition[automate.etats.indexOf(current)][automate.symboles.indexOf(word[i])];
                console.log("final -->" + current);
            }

            if (current == undefined) {
                //console.log("le symbole '" + word[i] + "' ne fait pas parti de l'aphabet")

            }
        }
        i++;
        console.log("index: " + i);
    }
    if (automate.etatsFinaux.indexOf(current) != -1) {
        automate.validAFN = true;
        console.log();
    }

    if (automate.validAFN) {
        console.log("mot valid");
    } else {
        console.log("mot invalid");
    }
}
```

AFD

create.html | script.js | style.css 1

Théorie des langages > Projet > js > script.js > testWord

```
601
602
603 function testWord(mot) {
604
605     let word = mot.toString().split('');
606     let current = automate.etatInitiaux;
607     console.table(current);
608     let i = 0;
609     let prevCurrent;
610     let character;
611     let posiCharacter;
612
613     try {
614         while (i < word.length) {
615             posiCharacter = i
616             if (current == "") {
617                 console.log("il n'y a pas transition sur " + prevCurrent + " avec le symbole " + character);
618                 testresult(mot, 3, prevCurrent, character, posiCharacter);
619                 break;
620             } else {
621                 prevCurrent = current;
622                 character = word[i];
623                 current = automate.tableTransition[automate.etats.indexOf(current)][automate.symboles.indexOf(word[i])];
624             }
625
626             if (current == undefined) {
627                 console.log("le symbole '" + word[i] + "' ne fait pas parti de l'aphabet")
628                 testresult(mot, 2, "", word[i], posiCharacter);
629             }
630
631             console.log(current);
632             i++;
633         }
634     } catch (error) {
635         console.log(error);
636     }
637     if (automate.etatsFinaux.indexOf(current) != -1) {
638         console.log("le mot est valid");
639         testresult(mot, 1, "", "", posiCharacter);
640
641     } else {
642         testresult(mot, 0, "", "", posiCharacter);
643     }
644
645 }
646
```