Rapport projet DSL n°2

Projet UXifier

Sujet: Quiz

Groupe F

Dina Abakkali - Sylvain Marsili - Thomas Martin - Clement Poueyto - Florian Striebel

Lien vers le code source

Dsl-21-22-f/quizz_ui : https://github.com/Dsl-21-22-f/quizz_ui

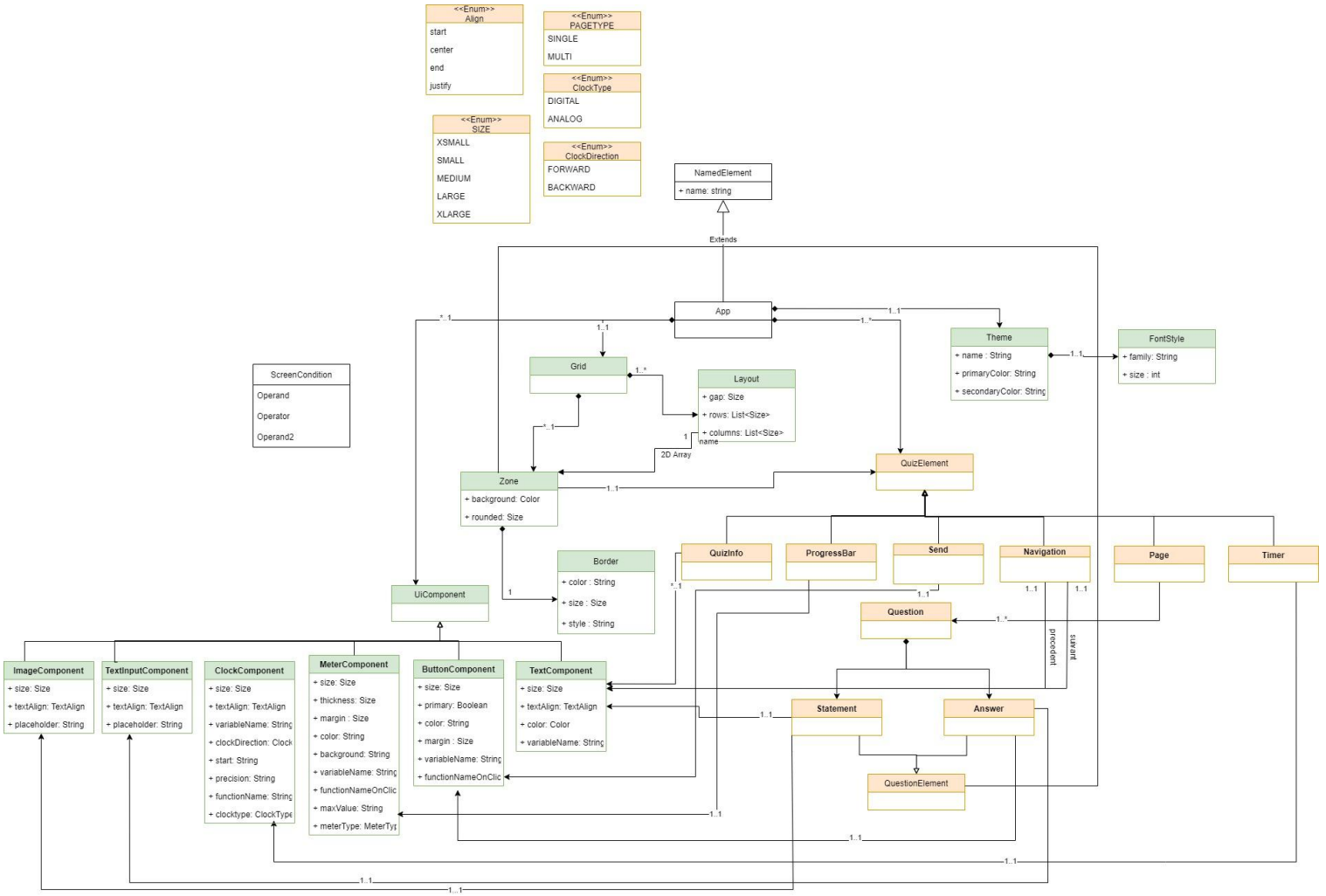
Sujet

Notre sujet concerne la définition d'un DSL permettant de créer une interface de quiz pour une web application. Cette interface affiche les éléments principaux d'un quiz, permet de jouer et de répondre aux questions posées.
Nous sommes partis sur un DSL externe car nous souhaitions former notre propre syntaxe afin de représenter tous les concepts possibles dans le cadre de la définition d'une interface web. Nous avons exploité cet aspect lors de la déclaration du positionnement de nos éléments dans l'interface et pour décrire nos composants. Nous avons choisi Antlr car nous l'avions utilisé dans le projet précédent et nous savions qu'il permettait de répondre à notre besoin pour ce projet.
Notre langage est textuel et est proche de notre domain model car nous sommes partis sur des fonctionnalités à implémenter pour en déduire une syntaxe.

Le but de notre DSL est de permettre au designer de créer différents types de quiz, il peut s'agir d'un quiz à choix multiple, à choix unique ou des réponses ouvertes. En effet, un quiz peut avoir la forme d'une évaluation, d'un jeu ou bien d'un test de personnalité.
C'est pourquoi, le designer peut décider s'il est possible de revenir en arrière ou non, et peut inclure une limite de temps etc…

Domain Model

Diagramme de classe du projet :



Dans ce modèle de domaine, nous avons représenté deux parties de notre système. La structure du métier du quiz (orange) et son affichage (vert).
Le métier du quiz est divisé en blocs qui regroupent les principales fonctionnalités d'un quiz que nous avons identifié comme :

- les informations principales
- la question et les réponses
- la progression du quiz
- le chronomètre
- la navigation entre les questions

Chacuns de ces blocs est placé dans une grille qui permet de déclarer la disposition des éléments sur une page.

Chacun de ces blocs correspond à un élément graphique de l'interface comme un bouton, un texte, une zone de saisie de texte etc…
Ces éléments graphiques sont modifiables via des propriétés comme la couleur et la taille que l'on retrouve sur la plupart des composants ou via des propriétés spécifiques comme la forme de la barre de progression.
Il est possible de définir des couleurs et une police d'écriture à appliquer par défaut via le thème.
Un élément d'un bloc peut prendre diverses formes comme dans le cas d'une question qui peut être sous la forme d'une image ou d'un texte et dans le cas d'une réponse qui peut être un bouton, un radio-bouton ou une saisie de texte.

Syntaxe concrète

BNF

```
grammar Quizz;


/*****************
 ** Parser rules **
 *****************/

root            :   declaration theme? grid EOF;

declaration     :   'application' name=IDENTIFIER;

theme           :   'theme' '{' ('primary color' ':' primary=(COLOR|HEX|SHADE) ('secondary color' ':' secondary=(COLOR|HEX|SHADE))?)?
 ('font' font)? '}';
font            :   '{' 'family' family=FONTFAM ('with' 'size' size=NUMBER 'px')? '}';

grid            :   zone+ layout+;
    zone        :   'zone' ':' name=IDENTIFIER ('alignment' alignement=ALIGN)? ('background' color=(COLOR|HEX|SHADE))?
    ('rounding' rounding=SIZE)?
    ('border' 'with' border )? (quizz_element)?;
    border      :   ('size' size=SIZE)? (',')? ('style' style=BORDERSTYLE)? (',')?
    ('color' color=(COLOR|HEX|SHADE))?;


layout          :   'layout' '{' screen_condition? gap? arrangement '}';
    screen_condition : 'when screen is ' media=MEDIA;
    gap         :   'gap' value=SIZE;
    columns     :   column+ ',';
    column      :   SIZE;

arrangement     :   'arrangement' '{' columns? line+ '}';
    line        :   row=SIZE? zone_name+ ',';
    zone_name   :   IDENTIFIER ;
```

```
quizz_element   : page | quiz_info | timer | progressbar | navigable | send;
    quiz_info   : ('title' 'with' title=text) ('description' 'with' description=text)?;
    page        : 'contains questions' question;
    question    : statement+ 'answer' 'with' answer+;
    navigable   :   'navigable forward' ('with' button) (backward)?;
    backward    :   'and backward with' button;
    send        :   'send quiz with' button;
    statement   : 'statement' 'with' (text_statement|picture_statement);
    answer      : single_answer | multiple_answer | open_answer;
    single_answer:  'single choice' 'with' button;
    multiple_answer : 'multiple choice' 'with' checkboxgroup;
    text_statement: 'text' 'with' text;
    picture_statement: 'picture' 'with' picture;
    open_answer : 'open' textInput;
    timer       : 'timer' 'with' clock;
    progressbar : 'progress' 'with' meter;


uiElement : button | text | clock | checkboxgroup | meter | textInput | picture;
    checkboxgroup : ('gap' gapanswer=SIZE)?;
    text :  'size' size=SIZE  (',' 'align' textAlign=ALIGN)? (',' 'color' color=(COLOR|HEX|SHADE))?;
    button : 'size' size=SIZE (',' 'color' color=(COLOR|HEX|SHADE))?  (',' 'margin' margin=SIZE)? (','''align' buttonAlign=ALIGN)?
    (','''label' labelValue=STRING)? (','''function 'function=STRING)?;
    clock :  (chrono=('chrono'|'countdown'))? (',' 'size' size=SIZE)? (',' 'align' textAlign=ALIGN)?
    (',' 'start' startTime=TIME)? (',' 'type' type=('DIGITAL'|'ANALOG'))?;
    meter :  'size' size=SIZE (',' 'type' type=('CIRCLE'|'BAR'|'PIE'|'SEMICIRCLE'))? (',' 'thickness' thickness=SIZE)?
    (',' 'color' color=(COLOR|HEX|SHADE))? (',' 'background' background=(COLOR|HEX|SHADE))?;
    textInput : 'size' size=SIZE (',' 'align' textAlign=ALIGN)? (',' 'placeholder' placeholder=STRING)?;
    picture         : 'height' height=SIZE ',' 'width' width=SIZE ;
```

```
/*****************
 ** Lexer rules **
 *****************/
//sPORT_NUMBER     :   [1-9] | '11' | '12';
NUMBER           :   [0-9]+;
TIME             : [0-2][0-3]':'[0-5][0-9]':'[0-5][0-9];
IDENTIFIER       :   LOWERCASE (LOWERCASE|UPPERCASE|NUMBER)+;
SIZE             :   'XXSMALL' | 'XSMALL' | 'SMALL' | 'MEDIUM' | 'LARGE' | 'XLARGE' | 'AUTO' | 'FULL';
BORDERSTYLE      :   'SOLID' | 'DASHED' | 'DOTTED' | 'DOUBLE' | 'RIDGE';
//QUESTION_UI     :   IMAGE   |   TEXT;
//IMAGE           :   'image' PATH;
TEXT             :   UPPERCASE (IDENTIFIER WS* NEWLINE)+;
//PATH            :   (.*|LOWERCASE|UPPERCASE)+;
//SPECIAL_CHAR    :   ':';

//SPECIAL_CHAR    :   ':'|'\\' |'/' |'.';
LETTER           :   (LOWERCASE|UPPERCASE);
CHAR             :   (NUMBER|LETTER);
HEX              :   '#' CHAR+;
COLOR            :   'WHITE' | 'BLACK' | 'RED' | 'BLUE' | 'GREEN' | 'YELLOW' | 'ORANGE' | 'GREY' |'VIOLET' | 'PINK' | 'BRAND';
SHADE            :   ('DARK' | 'LIGHT')'-'[1-6];
TYPE             :   'BUTTON'|'TEXT';
ALIGN            :   'CENTER'|'START'|'END';
MEDIA            :   'PHONE' | 'COMPUTER' | 'TABLET';
FONTFAM          :   'SERIF' | 'SANS-SERIF' | 'SCRIPT' | 'DISPLAY';
STRING           :   '\''~('\n'|'\r')*?'\'';
/*************
 ** Helpers **
 *************/

fragment LOWERCASE  : [a-z];                              // abstract rule, does not really exists
fragment UPPERCASE  : [A-Z];
NEWLINE             : ('\r'? '\n' | '\r')+      -> skip;
WS                  : ((' ' | '\t')+)           -> skip;     // who cares about whitespaces?
COMMENT             : '#' ~( '\r' | '\n' )*     -> skip;     // Single line comments, starting with a #
```

Les fonctionnalités

- Disposition des éléments sur la page :
    - création de zones, définition de leur taille et du positionnement

- Déclaration de la gestion du responsive :
    - définition d'une vue en fonction de la taille de l'écran (écran de téléphone, tablette ou ordinateur)

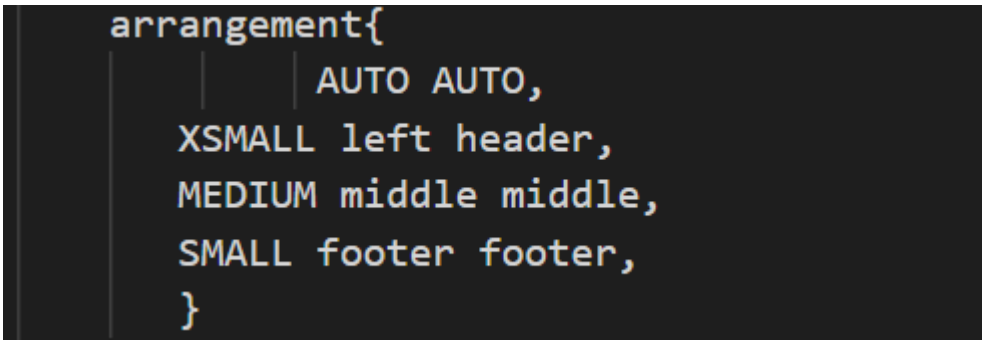
- Description du thème général de l'interface :
    - déclaration de la couleur primaire et secondaire du thème (couleur prédéfinie ou code couleur #50148C)
    - déclaration de la police du texte

- Description des blocs métier du quiz :
    - définition du type de réponses (QCM, simple, ouverte)
    - type de navigation (autoriser le retour à/passer la question ou non)
    - type de minuteur (chronomètre ou compte à rebours)

- Description de l'affichage des éléments du quiz :
    - Forme, couleur, bordure, taille, marge, alignement ou autres valeurs selon les éléments

La gestion de l’affichage des différents layouts

Pour l’affichage graphique nous avons voulu fournir à notre utilisateur un moyen visuel pour définir le positionnement des zones qu’il a créé. Nous nous sommes dit qu’il serait intéressant pour l’utilisateur de pouvoir placer ses composants dans un tableau pour représenter l’espace qu’ils prennent. Ci-dessous un exemple de la création de la disposition du layout.



La première ligne et la première cellule de chaque colonne représentent respectivement la taille de la colonne et la taille de la ligne. L’utilisateur peut ne pas renseigner la taille de toutes les lignes mais s’il renseigne une colonne il doit les renseigner toutes. La définition des layouts de cette façon permet à l’utilisateur en un coup d'œil d’avoir une idée d'où les composants sont placés et quelles sont leurs dimensions.
De plus, si l’utilisateur place une zone qu’il n’a pas définie cela renverra une erreur précise lui indiquant que cette zone est inexistante.

L’outillage

Pour permettre à l’utilisateur de développer son site de quiz plus efficacement nous lui permettons au travers de la commande “./quiz.sh” de pouvoir mettre à jour automatiquement le fichier “App.js” et de voir les résultats de ses modifications en temps réel, à chaque fois que celui-ci met à jour son fichier de quiz. Si celui-ci travaille sur plusieurs projets de quiz en même temps, il aura juste à ouvrir son nouveau fichier de quiz, et lorsque celui-ci sera sauvegardé automatiquement c'est le contenu du nouveau fichier de quiz qui sera généré dans App.js.
Pour que l’utilisateur ne soit pas pollué par les résultats des commandes lancée à chaque fois qu’il met à jour son fichier, nous affichons les résultats des commandes que lorsqu'il y eu une erreur dans la génération du fichier.
De plus, les fonctions implémentées ne sont pas écrasées car elles sont traitées dans le fichier “function.js” qui n’est pas régénéré à la compilation.

Les scénarios

Amandine veut pouvoir ajouter des thèmes et créer un quiz sur une seule page.
Amandine veut pouvoir adapter les quiz à la taille de l’écran de l'utilisateur.
Amandine veut pouvoir ajouter une navigation entre différentes pages de son quiz.
Amandine veut pouvoir créer un quiz avec uniquement des questions avec photo.

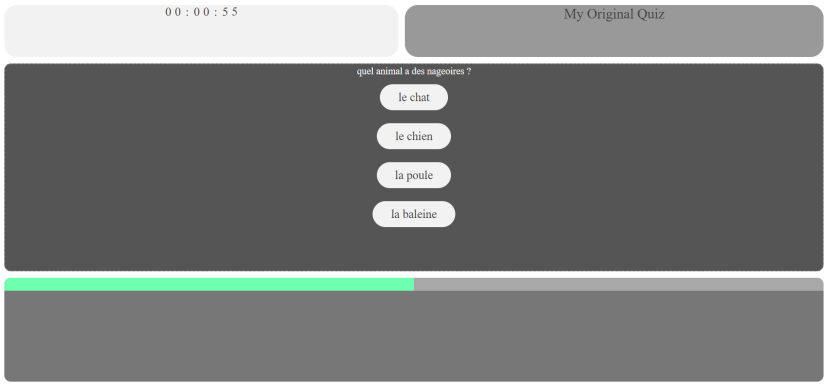
Comment utiliser l’outil de génération de quiz

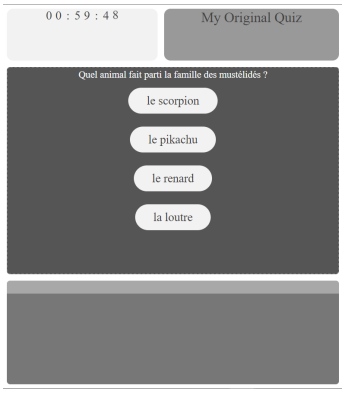
Configuration requise:

- Maven >1.8
- Java >1.8
- nodejs

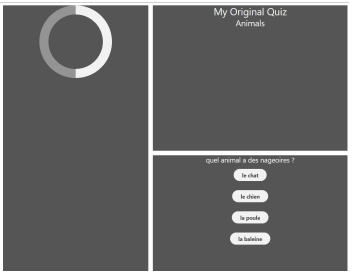
Une fois avoir récupéré le code sur le répertoire github. Il suffit de lancer le script run.sh. créer un fichier quiz dans le sous dossier "antlr\src\main\resources" celui-ci sera automatiquement détecté et lors de la sauvegarde le fichier sera généré dans App.js et les résultats de cette génération seront visible à l’adresse localhost:3000

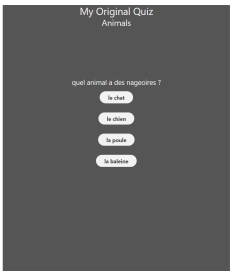
Exemples :

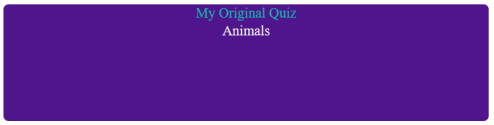

00 : 00 : 55
My Original Quiz
quel animal a des nageoires ?
le chat
le chien
la poule
la baleine

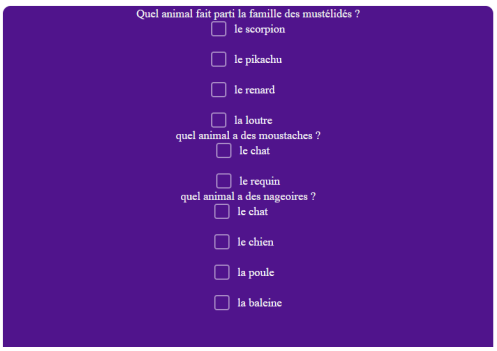

00 : 59 : 48
My Original Quiz
Quel animal fait parti la famille des mustélidés ?
le scorpion
le pikachu
le renard
la loutre

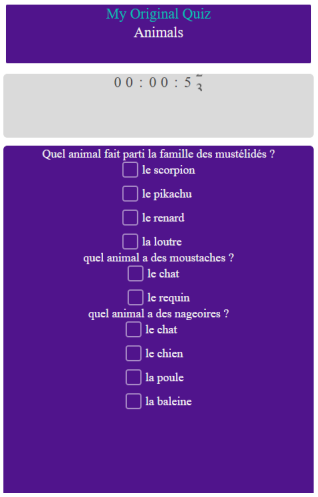

My Original Quiz
Animals
quel animal a des nageoires ?
le chat
le chien
la poule
la baleine

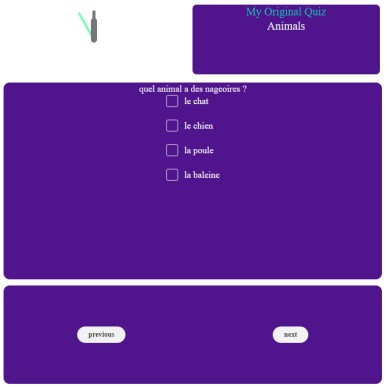

My Original Quiz
Animals
quel animal a des nageoires ?
le chat
le chien
la poule
la baleine


My Original Quiz
Animals
quel animal a des nageoires ?
le chat
le chien
la poule
la baleine


My Original Quiz
Animals


Quel animal fait parti la famille des mustélidés ?
le scorpion
le pikachu
le renard
la loutre
quel animal a des moustaches ?
le chat
le requin
quel animal a des nageoires ?
le chat
le chien
la poule
la baleine

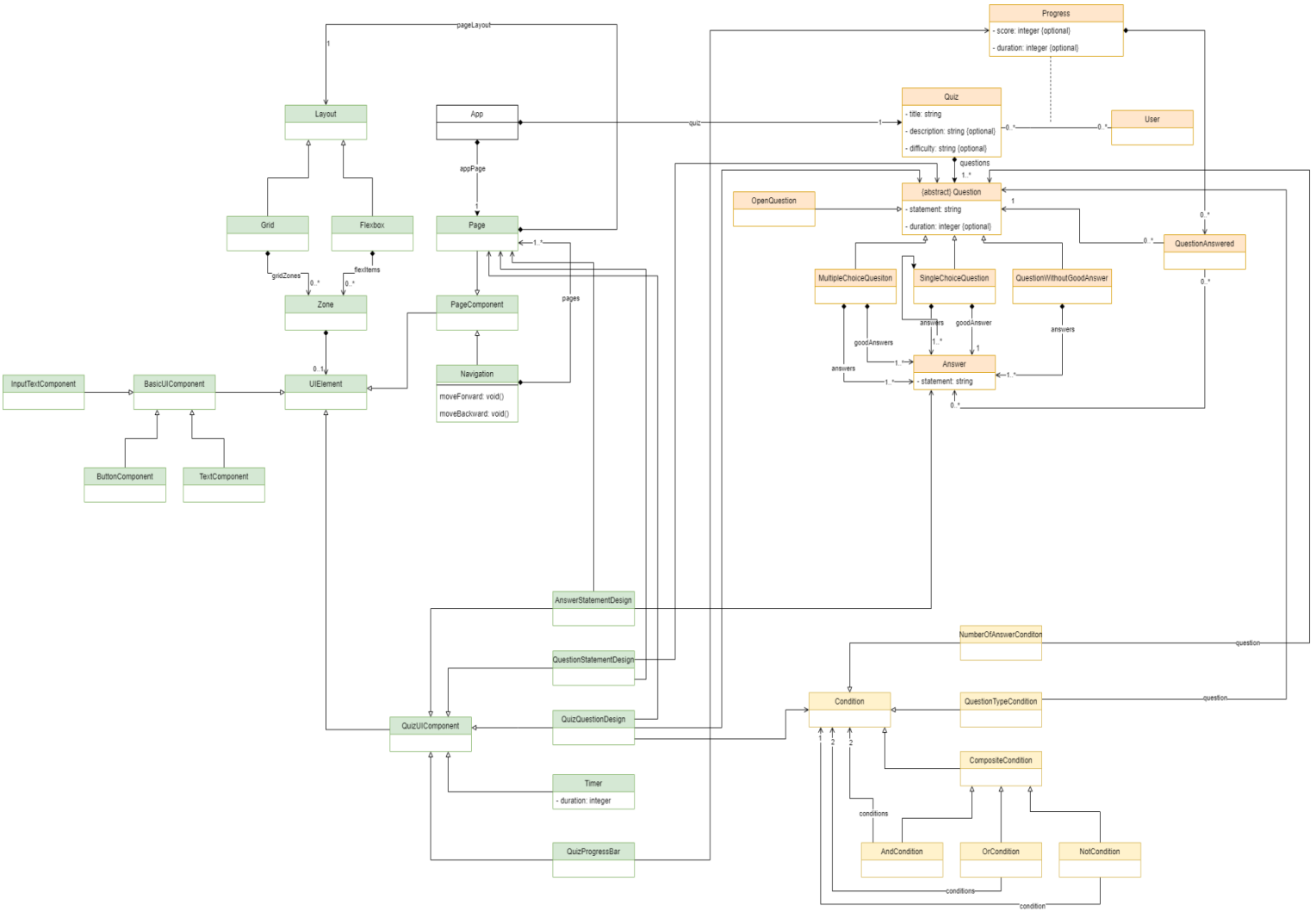

My Original Quiz
Animals
00 : 00 : 5
Quel animal fait parti la famille des mustélidés ?
le scorpion
le pikachu
le renard
la loutre
quel animal a des moustaches ?
le chat
le requin
quel animal a des nageoires ?
le chat
le chien
la poule
la baleine


My Original Quiz
Animals
quel animal a des nageoires ?
le chat
le chien
la poule
la baleine
previous
next

Version 2

L'idée de la deuxième version du projet débute par une nouvelle version du diagramme de classe. Nous avons réalisé ce nouveau diagramme pour rajouter la partie métier du quiz qui n'apparaissait pas sur le premier.

Nouveau diagramme de classe



A la vue du résultat, nous jugions notre première version trop complexe à la fois pour l'utilisateur et à la fois pour notre développement.

Au niveau de la grammaire nous avons trouvé que la manière de décrire le quiz était trop "technique", nous avons alors commencé par travailler sur le langage que l'on souhaiterait avoir. Ce dernier était volontairement plus proche d'un langage naturel, nous le voulions ainsi pour simplifier le travail du designer et le rendre accessible à tous.

BNF seconde version

```
grammar Quizz;

/*****************
 ** Parser rules **
 *****************/

root            :   declaration* appDeclaration  EOF;

appDeclaration  :   'Application' name=WORD 'from quiz' quizPath=STRING appAttributes;

appAttributes   :   (layoutAttribute themeAttribute) | (layoutAttribute themeAttribute);
layoutAttribute :   'uses layout' layoutName=IDENTIFIER;
themeAttribute  :   'uses theme' themeName=IDENTIFIER;

declaration     :   themeDeclaration | gridDeclaration | boxDeclaration;
themeDeclaration:   themeName=IDENTIFIER 'is theme with primary color' primary=COLOR'secondary color' secondary=COLOR'font family' fontFamily=FONTFAM;
gridDeclaration :   gridName=IDENTIFIER 'is grid with' gap=SIZE 'gap wich follows disposition' disposition;
boxDeclaration  :   boxName=IDENTIFIER 'is' (isFlex='flex')? 'box' ( 'with direction' direction=DIRECTION)? 'that contains' boxContent;

disposition     :   columns? row+;
columns         :   column+;
column          :   columnSize=SIZE;
row             :   rowSize=SIZE zone+;
zone            :   zoneName=IDENTIFIER;

boxContent      :   ((text | textInput | button | checkBox)+ | questions);
text            :   'text' (quizTitleBinding | 'with value' textValue=STRING) ('with font size' fontSize=NUMBER)? globalStyle?;
textInput       :   'text input' ('that contains place holder' textValue=STRING)? ('with font size' fontSize=NUMBER)? globalStyle?;
button          :   'button' ('that call' functionName=STRING)? ('contains value' textValue=STRING)?  globalStyle?;
checkBox        :   'checkboxgroup' ('that call' functionName=STRING)? ('and contains option' option=STRING)? 'with' ('gap' gapanswer=SIZE)? globalStyle?;
globalStyle     :   ',' ('aligned' textAlign=ALIGN)? ;
quizTitleBinding:   'binded to quiz title';

questions       :   'the questions:' navigable?;
navigable       :   'questions are navigable forward' (label)? (backward)?;
backward        :   'and backward' (label)?;
label           :   'with label' labelValue=STRING;
```

```
/*****************
 ** Lexer rules **
 *****************/

NUMBER          :   [0-9]+;
TIME            : [0-2][0-3]':'[0-5][0-9]':'[0-5][0-9];
IDENTIFIER      :   LOWERCASE (LOWERCASE|UPPERCASE|NUMBER)+;
SIZE            :   'XXSMALL' | 'XSMALL' | 'SMALL' | 'MEDIUM' | 'LARGE' | 'XLARGE' | 'AUTO' | 'FULL';
DIRECTION       :   'ROW' | 'COLUMN' | 'ROW_REVERSE' | 'COLUMN_REVERSE' | 'ROW_RESPONSIVE';
BORDERSTYLE     :   'SOLID' | 'DASHED' | 'DOTTED' | 'DOUBLE' | 'RIDGE';
TEXT            :   UPPERCASE (IDENTIFIER WS* NEWLINE)+;
LETTER          :   (LOWERCASE|UPPERCASE);
CHAR            :   (NUMBER|LETTER);
HEX             :   '#' CHAR+;
COLOR           :   'RED' | 'BLUE' | 'GREEN' | 'YELLOW' | 'ORANGE' | 'GREY' |'VIOLET' | 'PINK' | 'BRAND';
SHADE           :   ('DARK' | 'LIGHT')'-'[1-6];
TYPE            :   'BUTTON'|'TEXT';
ALIGN           :   'CENTER'|'START'|'END';
MEDIA           :   'PHONE' | 'COMPUTER' | 'TABLET';
FONTFAM         :   'SERIF' | 'SANS-SERIF' | 'SCRIPT' | 'DISPLAY';
STRING          :   '\''~('\n'|'\r')*?'\'';
WORD            :   LETTER+;

/*************
 ** Helpers **
 *************/

fragment LOWERCASE  : [a-z];                            // abstract rule, does not really exists
fragment UPPERCASE  : [A-Z];
NEWLINE             : ('\r'? '\n' | '\r')+      -> skip;
WS                  : ((' ' | '\t')+)           -> skip;     // who cares about whitespaces?
COMMENT             : '#' ~( '\r' | '\n' )*     -> skip;     // Single line comments, starting with a #
```

A partir de cette nouvelle grammaire, de ce nouvel objectif, nous nous sommes lancés dans le développement d'une nouvelle version car nous avons estimé trop coûteux d'effectuer tous ces changements sur le code existant. Nous pensions par ailleurs profiter de ce nouveau développement pour simplifier le comportement de notre dsl, son code, que ce soit au niveau du domain model ou au niveau d'antlr.

Comparaison

Avec la version 1 nous obtenons des fichiers comme cela

```
application exampleOne

    zone : middle
    background DARK-2
        rounding SMALL
      border with size SMALL, style DASHED, color DARK-3
      contains questions
      statement with text with size MEDIUM,  align CENTER
      answer with single choice with size LARGE, color LIGHT-2 ,margin SMALL

    zone : header
        background DARK-5
            rounding MEDIUM
        title with size XLARGE,  align CENTER

    zone : left
        background LIGHT-2
        rounding MEDIUM
        timer with countdown, size LARGE, type DIGITAL

    zone : footer
        background DARK-3
                rounding SMALL
        progress with size FULL, type BAR, thickness MEDIUM

zone: send
rounding SMALL
    send quiz with size SMALL, color DARK-4, margin AUTO, align START, label 'send'

layout {
    gap SMALL
    arrangement{
             AUTO AUTO,
       XSMALL left header,
       MEDIUM middle middle,
       SMALL send send,
       }
    }
```

Tandis qu’avec la version 2 nous obtenons des fichiers de ce type

```
# other layouts

questionList is flex box that contains the questions:
questions are navigable forward with label 'next' and backward with label 'previous'

send is box that contains text with value 'send' with font size 15 , aligned CENTER

middleLayout is grid with SMALL gap wich follows disposition
    AUTO questionList
    SMALL send

headerLayout is box that contains
    text binded to quiz title with font size 15 , aligned CENTER

# root layouts

rootLayout1 is grid with LARGE gap wich follows disposition
    XXSMALL headerLayout
    AUTO middleLayout

# theme

globalTheme is theme with primary color BLUE secondary color ORANGE font family SCRIPT

Application SinglePageGoal from quiz './quiz.json'
    uses layout rootLayout1
    uses theme globalTheme

```

On peut donc constater que nous avons essayé de diminuer au maximum l'aspect ‘json’ de notre grammaire en retirant les {} et les : . Comme dit précédemment, nous avons voulu avoir une grammaire plus proche d’un langage naturel, cela se ressent au niveau des déclarations qui sont sous forme de phrases.

Perspectives

- Nous avons deux branches de développement : la version 1 et la version 2. L'idée est d'amener la version 2 au même niveau que la première puis d'en poursuivre le développement.
- Avec du temps supplémentaire, nous souhaitons implémenter des actions/fonctions de bases que l'utilisateur final sélectionnerai pour les éléments du quiz directement dans le langage.
  ex : Lorsque l'utilisateur du quizz clique sur une bonne ou une mauvaise réponse :
    - passer à la question suivante
    - augmenter/diminuer le score
    - réinitialiser le temps
    - faire disparaître la réponse

- Ajouter un éditeur qui permet la coloration syntaxique et l'autocompletion avec antlr (e.g. Monaco-editor)

Implication des membre dans le projet

Dina

- Travail sur le diagramme de classe
- Définition de la grammaire d'une partie des composants
- Implémentation des composants UI en domain model ainsi qu'en Antlr

Clément

- Implémentation du Layout en Grille
- Développement de la structure du projet
- implémentation des éléments de Quiz (question/réponses, chronomètre, titre, barre de progression) et les composants de UI côté domain model et syntaxe.
- Diagramme de classe

Thomas

- Personas et user stories
- Nouveau diagramme de classe
- Input text dans la première version
- Proposition d'une nouvelle grammaire et implémentation de la deuxième version pour y aboutir (en suivant également la logique de nouveau diagramme de classe)
    - Refactor à partir de 0 pour tenter de corriger la complexité de la première version

Sylvain

- Gestion de la présentation des questions (tout sur une page, ou navigation entre questions, fonction et button pour la fin d'un quiz) dans le kernel et la grammaire.
- Quelque automatisation sur la grille (size non obligatoire)
- Développement de nouveaux éléments pour la nouvelle proposition de refactor de la grammaire et du kernel..
- Gestion de l'affichage selon type de questions

Florian

- Implémentation de l'outil de compilation du langage avec gestions des erreurs
- Ajout de la notation en table pour définir la grille d'un layout
- Ajout de modification pour rendre l'affichage plus customisable (customizable border, rounded box)
- ajout du responsive sur les layouts