Class Exercise\_Definition

Skládá se z Definition\_MetaData a položek, které příklad určují:

## Definition\_MetaData:

* Language initialLanguage // default english, autocomplete one from given options.
* ExerciseType type // default numerical, autocomplete one from ogiven options.
* String title // Textfield, must be nonempty. Různý od všech ostatních existujících titulků v daném jazyce.
* String description // Textfield, must be nonempty.
* List<Topic> topics // Na začátku prázdné. Musí se vybrat alespoň jedno téma. Autocomplete from options. V UI se vybraná options zobrazují jako chips, která jde z listu odstranit.
* List<Grade> grades // Multicheckbox 1st – 9th grade. Musí být vybrána alespoň 1, na začátku žádná.
* Bool autoGenerateThumbnail // switch, default on.
* Image image to be loaded from local machine. -> editor will save it at apropriate place and save string filePath only to exercise metadata.

## Kontroly při validaci inputu metaDat:

1. Language je z množiny povolených jazyku. (enum) V Blazor serveru bude list hardcoded jako list.
2. ExerciseType je z množiny povolených typů příkladů.
3. Titulek musí být neprázdný string.
4. Titulek musí být různý od všech ostatních známých titlků v daném jazyce.
5. Popisek musí být neprázdný string.
6. Témata jsou neprázdný seznam.
7. Každé téma je v daném jazyce známé téma.
8. Třída je vybrána alespoň 1.
9. Pokud je autogenerate obrázku vypnutý, byl nahrán korektní obrázek.
10. Definice korektního obrázku přijde později od web dev týmu.

## Build up entit:

**Class Variable:**

* VariableType type // from: { int Set, int Range, string, double, operator }, vyber prave jednoho -> radio group.
* String name // text field, must be nonempty starting with characters: [a-zA-Z] and must be unique for all variable names. Also default naming appears naming all non operator variables A, B, C, … And naming all operator variables op1, op2, op3, …
* For int range only:
  + Int min, max, increment // min must be at most max, increment must be at least one.
* For all others:
  + All elements of the set. // must be nonempty set.

**Class MacroText:**

* Can be either child class Macro or child class ConstText // switch
* Macro requires string such that it matches a name of existing variable.
* Text requires some nonempty string.

## Kontroly při validaci inputu Variable a MacroTextu

1. Variable má vybraná práve jeden typ.
2. Je-li typ int range
   1. Min, max, increment je definovaný
   2. Min je nanejvýš max
   3. Increment je alespoň 1
3. Jinak pro ostatní typy
   1. Množina prvků variable je neprázdná
4. Jméno Variable je neprázdný string.
5. Jméno Variable začíná na nějaký z characteru: [a-zA-Z]
6. Jméno Variable obsahuje právě jenom znaky: [\_a-zA-Z0-9]
7. Jméno každé Variable je unikátní.
8. Je-li MacroText ConstText, je text neprázdný string.
9. Je-li MacroText Macro, potom string macra se shoduje se jménem nějaké proměnné.
10. Každá Variable je odkázána alespoň jedním makrem někde. (zadání, otázky atd.) Tj. Variable má v příkladu roli, není redundantní.

# Položky určující podobu příkladu

Oba typy následujících metod jsou interně reprezentované jako List<string>. Navíc každá obsahuje List<string> comments, kde se povinně v lidském jazyce dokumentuje co kod počítá/kontroluje pro dokumentaci pro budoucí editory příkladu. Obě metody také mají parametry nějakou konkrétní isntanci Varibles, tj. počítají stav konkrétní varianty příkladu.

**Buď ResultMethod** metoda vracející datový typ ResultType. **Buď ConstraintMethod** metoda vracející bool.

Je-li příklad typu početní příklad potom:

* Questions.Count = 0
* Má právě jednu metodu v results.

Je-li příklad typu slovní úloha potom:

* Má alespoň 1 question.
* Právě tolik results kolik questions.

Povinné položky jsou:

* Variables -> Alespoň 1
* Assignment s alespoň jednou položkou v MacroText.
* resultType
* results

Volitelné položky jsou:

* constraints
* solutionSteps
* imagePaths

**Položky v definici:**

* List<Variable> variables // Alespoň jedna Variable
* MacroText assignment // Neprázdný seznam
* List<MacroText> questions // Neprázdné pro slovní úlohu, prázdné pro početní příklad
* ResultType resultType // Výběr právě jednoho
* List<ResultMethod> results // Počet result metod musí být stejný jako počet questions, každá result metoda se zkompiluje – roslyn analyzer je happy.
* List<ConstraintMethod> constraints // každá constraint je korektní, tj. zkompiluje se, tj. roslyn analyzer je happy.
* List<MacroText> solutionSteps // volitelné, může být Count = 0
* List<string> imagePaths // ! různé od cest k thumbnail, tyto obrázky jsou součástí zadání. Obrázku může být 0 až velikost paměti počítače.

## Kontroly při validaci inputu položek v definici

1. Alespoň jedna proměnná je dobře definovaná.
2. Macro text má alespoň jednoho člena.
3. Je-li příklad typu početní questions.Count = 0.
4. Je-li přklad typu slovní úloha je questions.Count > 0
5. resultType je vybraný právě 1.
6. Je-li příklad typu početní results.Count = 1.
7. Je-li příklad typu slovní úloha results.Count = questions.Count
8. Každá ResultMethod je korektní a zkompiluje se, tj:
   1. Každá použitá proměnná je buď inicializovaná uvnitř nebo referencuje jméno nějaké Variable.
   2. String představuje korektní C# syntax, který Roslyn akceptuje.
   3. Metoda vrací datový typ resultType.
9. Každá ConstraintMetoda je korektní a zkompiluje se podle analogické logiky jako ResultMetoda.
10. Pokud uživatel u některé kterékoli metody zaškrtne codeDefined = false, potom pro ni bod 8 a 9 platit nemusí. (Předpokládá se, že potom kód dodefinuje někdo jiný, asi programátor, později)
11. Každý solution step obsahuje odkazy na existující variables.

!! Mindnote: Pro ResultsMethod a ConstraintMethod zvážit možnost import z nějakého souboru, případně i možnost předvygenerovaní source file s proměnnými, kam lze dopsat kod, tak abychom při psaní měli komplet servis VS. (metody budou většinou krátké)

!! Jak upravit definici solution step, aby šel psát C# kod? (nutné i pro středně komplexní příklady, natož ty koplexní.)