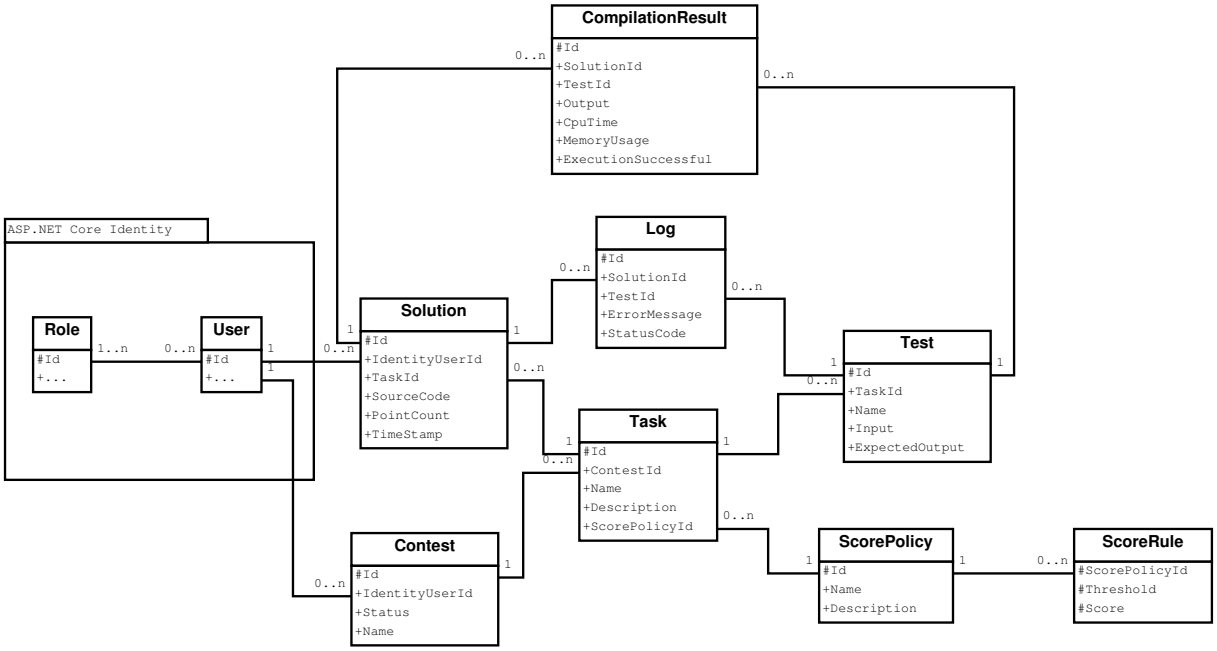
Algenic – serwis z konkursami algorytmicznymi

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Elektrotechniki, Automatyki,
Informatyki i Inżynierii Biomedycznej

Kacper Tonia Sławomir Kalandyk Mateusz Ruciński

1 Baza danych



Rysunek 1: Diagram obecnej bazy danych

Baza danych jest generowana za pomocą narzędzi dostarczanych przez Entity Framework Core. Przy uprzednim utworzeniu modelu oraz klas konfigurujących własności tabel (n.p. klucze obce), można wygenerować migrację funkcją add-migration i utworzyć lub dokonać zmian w istniejącej bazie danych funkcją update-database. Obie te czynności muszą być wykonane z poziomu konsoli.

W razie wszelkich błędów przy aktualizacji bazy danych, istnieje możliwość ręcznego modyfikowania migracji.

2 Technologia i narzędzia

Język programowania: C#
Framework: ASP.NET Core 2.2
System kontroli wersji: Git

Platforma: GitHub
Serwis do testów automatycznych: Travis CI
Framework do testów jednostkowych: xunit
Framework do testów funkcjonalnych: Selenium WebDriver
Zdalna kompilacja: JDoodle

Projekt został oparty o framework ASP.NET Core ze względu na jego popularność, wieloplatformowość i fakt bycia oprogramowiem typu open source. Zrezygnowaliśmy z sięgnięcia po najnowszą wersję frameworka, 3.0, na rzecz bardziej dojrzałej i potencjalnie stabilniejszej wersji 2.2.

Rozważyliśmy wykorzystanie usługi Azure, co pozwoliłoby na skupienie większości aspektów pracy nad programem w jednym miejscu (począwszy od przechowywania plików projektu, poprzez zarządzanie bazą danych projektu, skończywszy na automatycznym testowaniu zmian i wdrażaniu aktualnej wersji aplikacji w chmurze Azure). Stosunkowo długie czasy testowania i wdrażania w chmurze, a także względnie niewielka złożoność naszego projektu sprawiły, że zdecydowaliśmy się na uruchamianie i testowanie aplikacji lokalnie, wraz z lokalną bazą danych w technologii SQL Server.

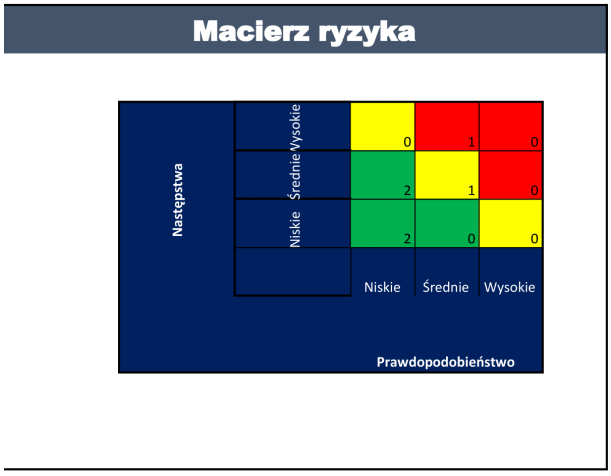
Ostatecznie wybraliśmy platformę GitHub m.in. ze względu na wbudowane narzędzia do zarządzania projektem, takie jak tablica zadań, które ułatwią systematyzację zadań związanym z etapem implementacji i przydzielanie owych zadań członkom zespołu. Jako element praktyki *Continuous Integration*, zmiany dodawane do głównego repozytorium są automatycznie testowane na platformie **Travis CI**.

Na późniejszym etapie prac nad projektem możemy ponownie rozważyć wykorzystanie Azure, chociażby w celu uruchomienia wspólnej, trwałej bazy danych w chmurze.

W celu kompilowania kodów przysyłanych przez użytkowników wykorzystamy serwis **JDoodle**, oferujący REST-owe API do zdalnej kompilacji. Kompilowanie niezaufanego kodu na serwerze niosłoby poważne zagrożenie, o ile nie podjęlibyśmy specjalnych środków bezpieczeństwa. Użycie istniejącej usługi jak JDoodle redukuje potencjalne problemy z bezpieczeństwem.

W darmowym wariancie JDoodle pozwala na przeprowadzenie ok. 200 kompilacji dziennie. Może być to zbyt mała wartość podczas intensywnego testowania, jednak prostota użycia i bogata ilość dostępnych języków programowania przekonuje nas ostatecznie do owego rozwiązania.

3 Analiza ryzyka



Rysunek 2: Macierz ryzyka

Macierz ryzyka tworzymy w celu określenia poziomu ryzyka związanego z produkcją i działaniem produktu w celu obniżenia negatywnego wpływu ryzyka na funkcjonowanie danego podmiotu i podejmowanie odpowiednich działań służących przeciwdziałaniu i ograniczaniu ryzyka. Po stworzeniu listy ryzyk i oszacowaniu ryzyka dla każdego elementu z listy można wywnioskować, że najwyższe ryzyko jest związane z błędem ludzkim (programisty). Minimalizować to ryzyko możemy, testując oprogramowanie przed wdrożeniem oraz dbać o obecność backupów na wypadek problemów z aktualizacją.

	Ryzyko	Prawdopodobieństwo	Następstwa	Poziom ryzyka	Sposób zapobiegania	skala	wartość
1	wadliwy kod przsłany przez użytkownika	3	0	niskie	korzystanie z JDoodle	0	brak
2	błędy oprogramowania	6	9	wysokie	praca lokalnie, testy	3	niskie
3	problem z serwerem	6	6	średnie	korzystanie z Microsoft Azure	6	średnie
4	podłączenie sie cudzego złośliwego oprogramowania	3	6	niskie	korzystanie z Microsoft Azure	9	wysokie
5	problem z połączeniem z JDoodle	3	3	niskie	ze względu na niskie ryzyko zamiast zapobiegać zdecydowaliśmy się obsłużyć te przypadki		
6	problem z połączeniem z bazą danych	3	3	niskie			
7	zmiana polityki JDoodle	3	6	niskie	napisanie programu w ten sposób aby w łatwy sposób móc zmienić serwis dostarczający nam funckjonalność JDoodle		

Rysunek 3: Lista ryzyk

4 Działania użytkowników

4.1 Wspólne

- Każdy użytkownik ma możliwość zarejestrowania się w aplikacji i zalogowania
- Każdy użytkownik może zobaczyć aktualnie trwające lub zakończone konkursy

4.2 Podstawowy użytkownik

- Może dołączyć do konkursu
- Po dołączeniu do konkursu może wyświetlać znajdujące się w nim zadania i przesyłać rozwiązania do nich

4.3 Egzaminator

- Posiada możliwości podstawowego użytkownika
- Może tworzyć konkursy
- Po utworzeniu konkursu może dodawać do niego zadania
- Po dodaniu zadania może dodawać do niego testy
- Ma możliwośc zmiany stanu konkursu (n.p. po dodaniu zadań może go wystartować, a później zakończyć)
- Uruchamia kompilację rozwiązań zadań konkursowych po zakończeniu konkursu (WIP)
- Dodaje listę par postaci (próg procentowy, ilość punktów), według której ocenianie jest zadanie (WIP)

4.4 Administrator

- Posiada możliwości podstawowego użytkownika oraz egzaminatora
- Może zmieniać role innych użytkowników (WIP)