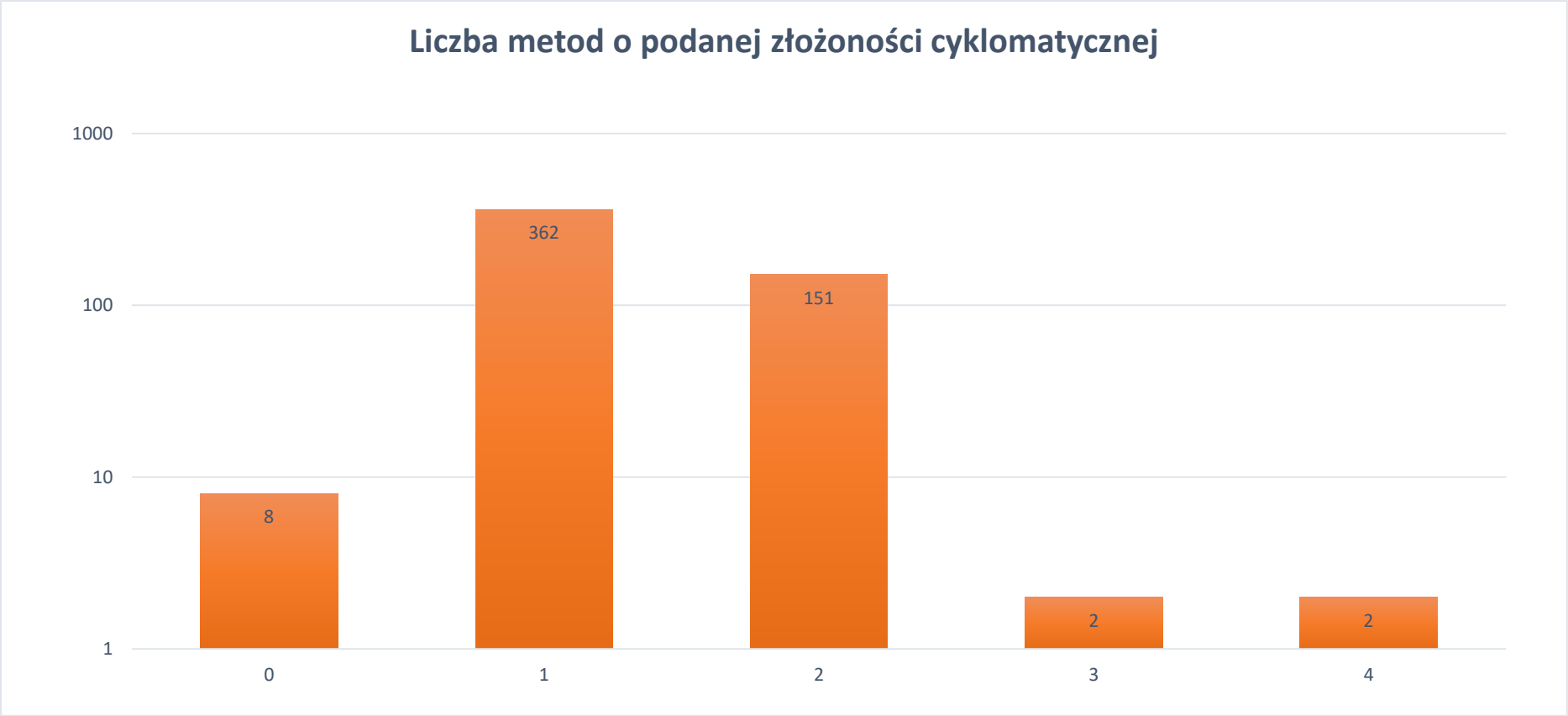
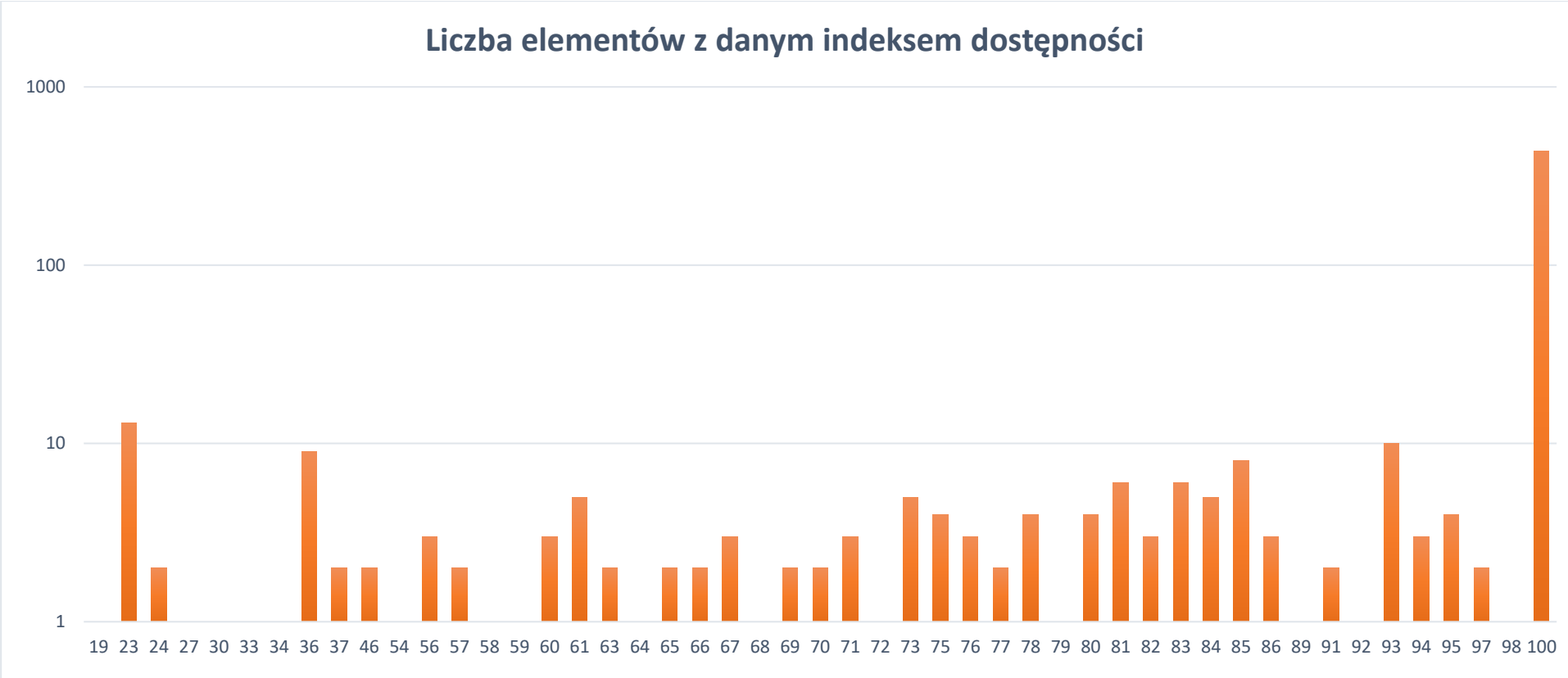


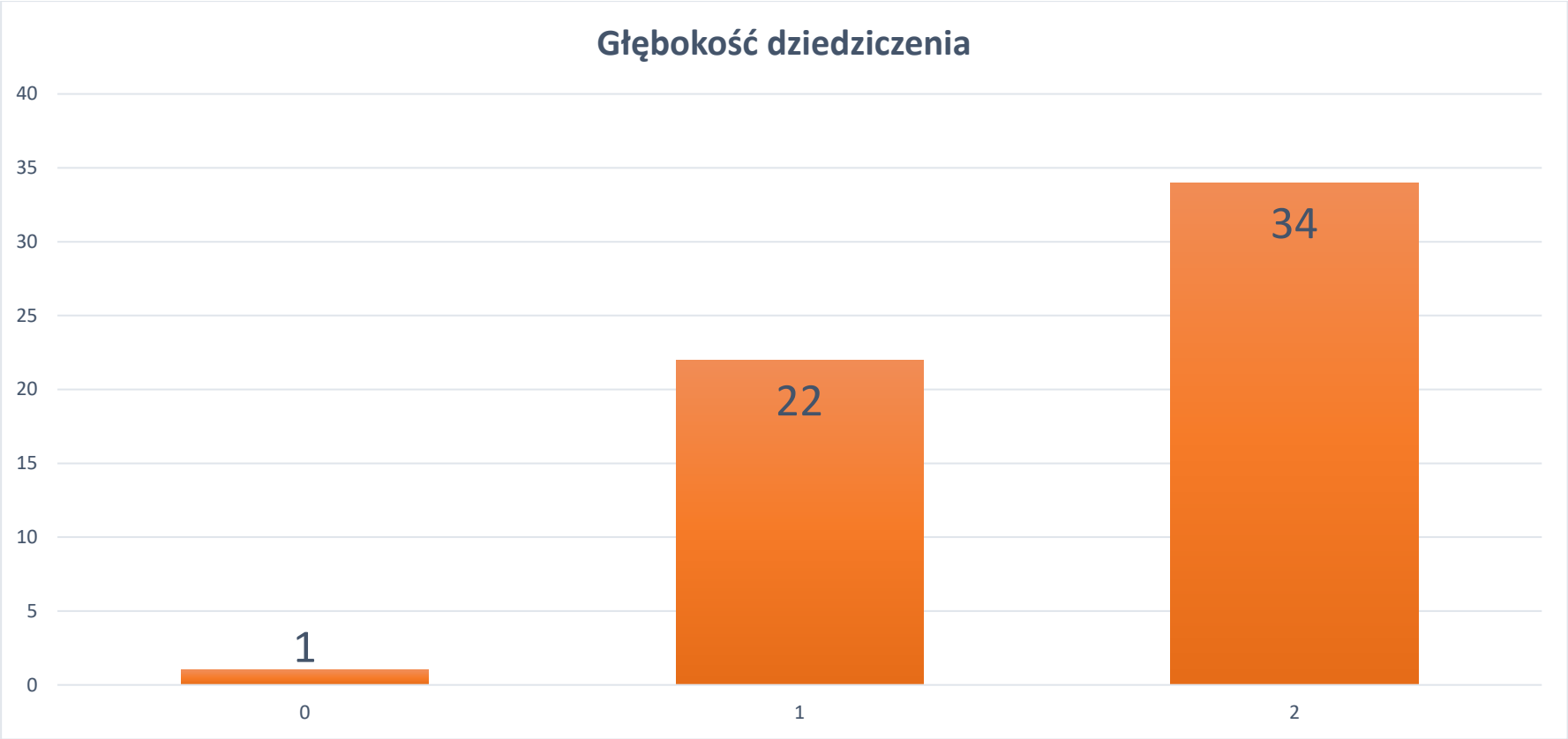
Złożoność cyklomatyczna to metryka oprogramowania używana do pomiaru stopnia skomplikowania programu. Podstawą do wyliczeń jest liczba dróg w schemacie blokowym danej metody, co oznacza wprost liczbę punktów decyzyjnych. Akceptowalne wartości dla złożoności cyklomatycznej metod zawierają się w zakresie [0, 10].

Wniosek: stworzony kod jest prosty i obarczony niskim ryzkiem.



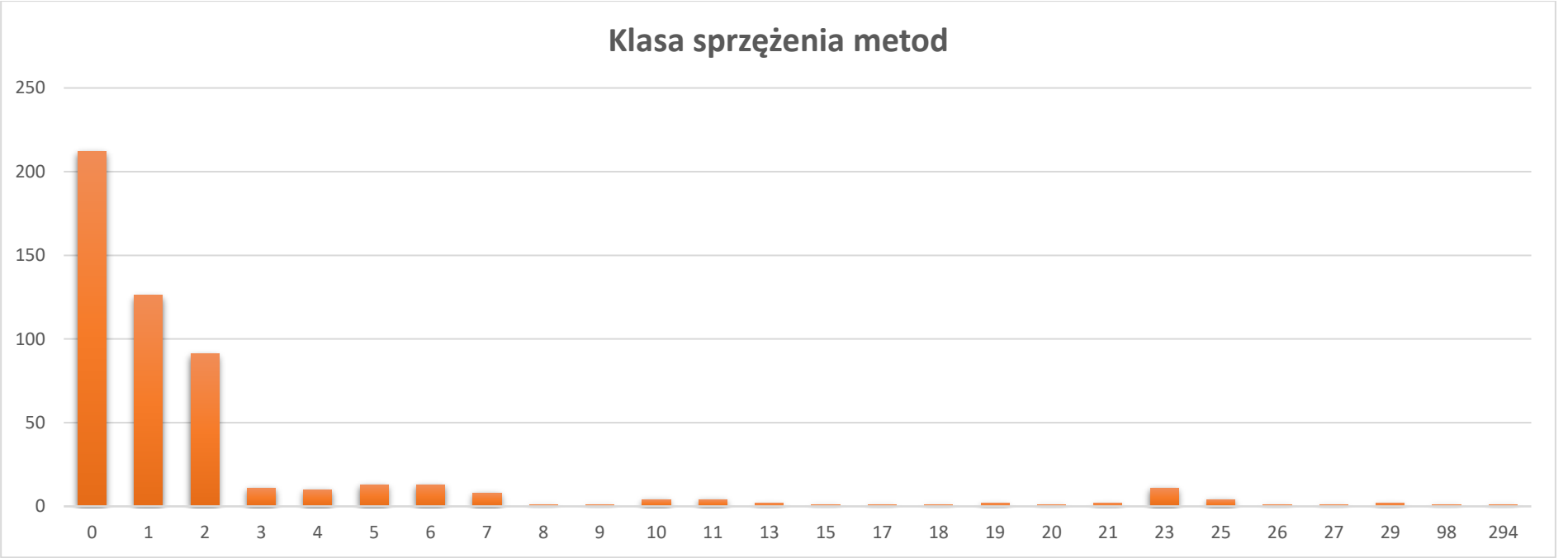
Indeks dostępności (łatwości utrzymania) reprezentuje względną łatwość utrzymania kodu. Im wyższa wartość tym łatwiejsza konwersacja. Uznaje się, że wartości od 20 do 100 stanowią zakres najlepszego pod tym względem kodu.

W stworzonym projekcie metody o najniższym indeksie są związane z migracjami, których kod jest generowany automatycznie, a także z wstawianiem początkowych wartości do bazy danych (tutaj, jednak dane generowane są przez stworzony przez nas projekt pythonowy, dlatego utrzymanie nie stanowi istotnego problemu (usunięcie czy edycja pola w bazie to najczęściej zmiana jednej linijki kodu, co więcej w wersji produkcyjnej programu generowanie danych startowych zostało by znacznie ograniczone (obecnie jest ono rozbudowane dla testów i prezentacji).



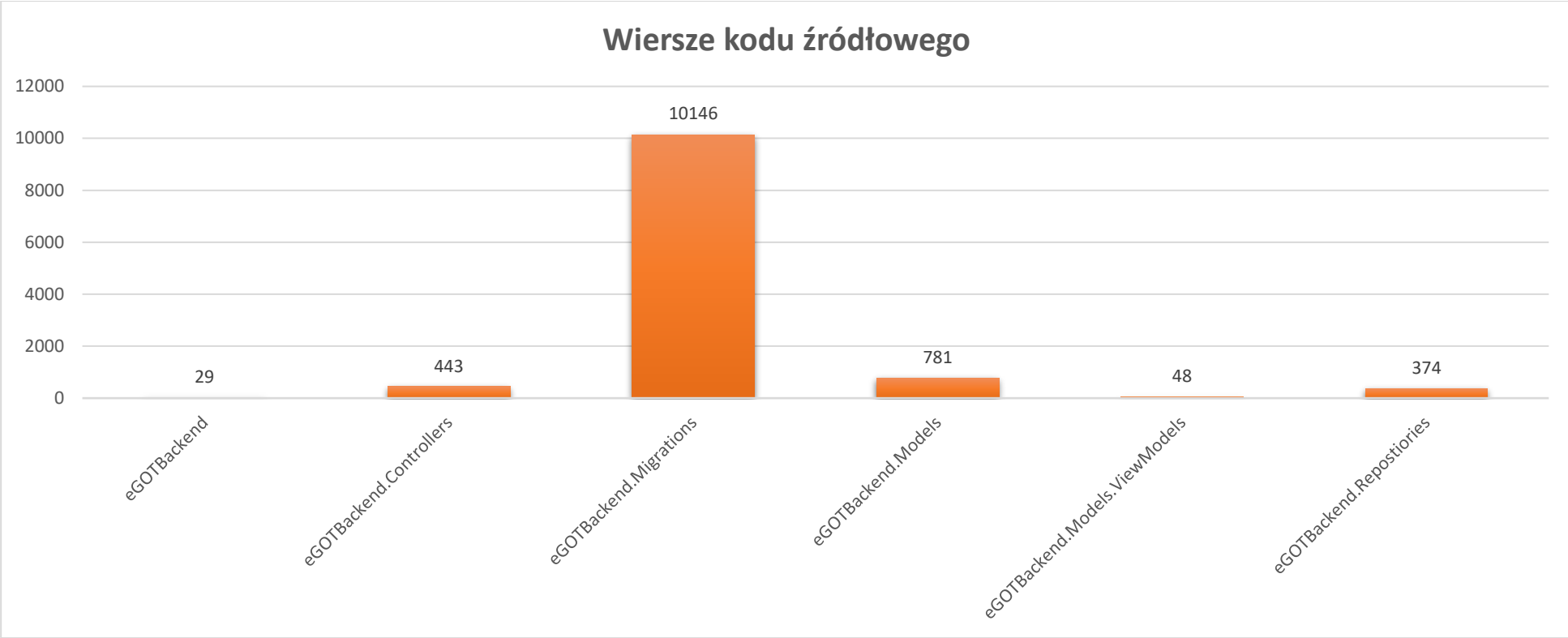
Wskazuje na liczbę różnych klas, które dziedziczą po sobie, aż do klasy bazowej. Im większa wartość tym większa szansa na potencjalny destruktywny wpływ zmian w klasie bazowej na metody po niej dziedziczące.

Maksymalna głębokość dziedziczenia w naszym projekcie to 2, dzięki temu opisany wyżej problem jest mało prawdopodobny.



Mówi o sprzężeniu pomiędzy unikatowymi klasami poprzez parametry, zmienne lokalne, zwracane typy, wywołania metod, instancje, klasy bazowe czy interfejsy. Dobre oprogramowanie powinno odznaczać się, jak największą spójnością z zachowaniem, jak najmniejszego sprzężenia.
Przyjęto, że ostrzeżeniem dla programisty jest sprzężenie metody powyżej 30.

W naszym projekcie znajdują się dwie metody, których sprzężenie przekracza liczbę 30:
- ApplicationDbContext.OnModelCreating(ModelBuilder) [294] – złożoność tej metody jest zależna od ilości encji oraz ich stopnia skomplikowania (liczba pól czy ograniczeń). Jej skomplikowanie nie stanowi problem ze względu na to, że jej kod generowany jest automatycznie na podstawie modelu bazy danych. Jest za to odpowiedzialny “entityframework” i jego polecenie Update-Database.
- eGOTBackend.Migrations.mountainsystemadded.Up(MigrationBuilder) [98] – metoda ta została również wygenerowana przez „entity framework” i jego polecenie Add-Migration. Użytkownik nie powinien zmieniać ręcznie treści tej metody, gdyż generowana jest ona automatycznie na podstawie zmian w modelu tak, by utrzymywać bazę danych aktualną i zgodną z modelem w porojekcie ASP.NET. Sprzężenie tej metody nie stanowi zatem problemu.
W związku z powyższym nie ma konieczności rozbijania tych metod na mniejsze w celu zmniejszenia sprzężenia.



Migracje stanowią największą część linii kodu. Natomiast pozostałe elementy (Controllers, Models, ViewModels, Repositories) zawierają łącznie około 1500 lini kodu.

Biorąc pod uwagę skomplikowanie modelu bazy danych można stwierdzić, że zastosowana technologia ASP.NET nie wymaga dużej liczby programistów i czasu do stworzenia aplikacji webowej.

Oprócz kodu ASP.NET zrealizowane zostały projekty obsługujące testy jednostkowe (Selenium) oraz generator danych (Python). Te projekty nie są jednak rozbudowane, nie zawierają znacznego dziedziczenia oraz sprzężenia ze względu na swoją charakterystykę, dlatego też zostały pominięte w rozważaniach dotyczących analizy metryk kodu.

Jeżeli chodzi o aplikację frontendową w React:

Złożoność cyklomatyczna o wartości co najwyżej 10 została zapewniona przez eslint i ustawioną w nim zasadę:

```
"rules": {
    "complexity": ["error", 10]
}
```

Liczba linii kodu to odpowiednio dla głównych folderów: 71 – Commons (stałe, ścieżki), 331 – Components (pliki .jsx i .css często używanych elementów, takich jak lista z paginacją), 2096 – Views (pliki .css i .jsx wszystkich widoków, które widzi użytkownik).

Pozostałe metryki nie są kluczowe w projekcie React:
- W React dziedziczenie nie jest preferowanym sposobem rozszerzania komponentów (na stronie Reacta można znaleźć informację, że w tworzeniu Facebooka został wykorzystany jedynie mechanizm dziedziczenia po React.Component)

- Metody nie są skomplikowane, kod stanowi głównie html, pliki stylów i logika javascriptowa (głównie obsługa zapytań HTTP).