1. Struktura aplikacji Ruby on Rails

Tworzymy nową aplikację Ruby on Rails

```
$ cd katalog-sklonowany
$ gem install rails bundler --no-ri --no-rdoc
$ rails -h #pokaże nam dostępne opcje podczas tworzenia nowej aplikacji
$ rails new . --skip-spring --skip-test --skip-coffee
$ atom .
```

Sprawdzamy czy działa

```
$ rails s
```

albo

```
$ bundle exec rails s
```

Struktura aplikacji

```
app/assets - pliki js, css, obrazki
app/controllers - kontrolery (mvC)
app/helpers - używane w widokach jeśli potrzeba logiki
app/models - modele (Mvc)
app/views - widoki i layouty (mVc)

/config - pliki konfiguracyjne
/db - migracje, seedy

Gemfile - plik z gemami
Gemfile.lock - autogenerowany plik z wersjami zainstalowanych gemów
```

2. Modele

Generowanie nowego modelu

```
$ rails g model NazwaModelu nazwapola:typ nazwapola:typ
$ rails g model User name:string
```

```
$ rails db:migrate
```

3. ActiveRecord

Manipulowanie obiektami poprzez ORM

```
$ user = User.new(name: 'Jan Kowalski')
$ user.save
$ User.create(name: 'Jan Kowalski')
$ user = User.create(name: 'Janina Kowalska')
$ user.update(name: 'Katarzyna Kowalska')
$ user.destroy
```

Najczęstsze operacje do wyciągania obiektów z bazy

```
$ User.find(1)
$ User.find_by(name: 'Jan Kowalski')
$ User.last
$ User.last(5)
$ User.first
$ User.all
$ User.count
$ User.where(name: 'Jan Kowalski')
```

4. Asocjacje

```
$ rails g model Car user:references brand:string
```

Należy pamiętać żeby uzupełnić w modelu relację:

```
class User
  has_many :cars
end
```

Relacja w modelu `Car` uzupełnia nam się automatycznie

5. Walidacje

https://guides.rubyonrails.org/active_record_validations.html

6. Work work work work

1. Zrobić model Message z polem `content` typu string i `author` typu string
2. Stworzyć w konsoli 5 obiektów message'a z różnymi wartościami
3. Stworzyć model Comment z polem `content` (string) i `author` (string) który będzie miał relację do Message (`message has_many :comments`)
4. Dodać walidację na pola `content` (presence) oraz walidację na ilość znaków (max. 140 w obu)
5. Napisać pętlę, która przeiteruje się przez `message` i wyrzuci na konsolę ich autorów. (można użyć `puts`)
6. Napisać query, które wyciągnie z bazy wszystkie wiadomości użytownika o podanym `name` .
7. (trudne) Stworzyć jeszcze jeden obiekt message, którego pole `created_at` jest ustawione na datę jutrzejszą. Napisać query, które wyciągnie z bazy wszystkie wiadomości, które zostały stworzone wcześniej niż 5 minut temu i mają więcej niż 100 znaków. (tip: sprawdź jak działa `Date.today + 120.minutes` i dostosuj do zadania)
8. Przeczytać:
   `https://guides.rubyonrails.org/association_basics.html`
   `https://guides.rubyonrails.org/active_record_validations.html`
   `https://guides.rubyonrails.org/active_record_basics.html`
   `https://guides.rubyonrails.org/active_record_querying.html`

W przypadku pytań i eksperymentowania w domu piszemy na slacku na kanale `#pomoc` . Zaglądamy tam więc będziemy starali się odpowiadać i poradzać.