Design Document

High Speed Rail

Andy Gergel | E.J. Schroeder | Matt Moellman | Mikael Soto

Table of Contents

Classes

Question

The primary basis for discussion among users, each instance of the Question class has the following attributes:

- title - a short descriptive phrase to display to users in the question index
- content - the fully detailed question itself
- created_at - timestamp detailing when the question was created
- updated_at - timestamp detailing when the question was last updated

These attributes allow us to display the required information to the user when choosing to view a question in detail. The Question class validates the presence of both the title and content so that there cannot be empty questions.

Answer

A response post to a question, used within each Question's view. Each Answer has these attributes:

- `content` - the text that a user submitted as a response
- `question_id` - a reference to the question that the answer is a response to
- `created_at` - timestamp detailing when the answer was created
- `updated_at` - timestamp detailing when the answer was last updated

Each answer validates the presence of its content so that no answers may be submitted as an empty response.

Rating

For storing a rating to an Answer. Each instance represents a single rating to an Answer. A rating has the following attributes:

- `answer_id` - a reference to the question that this rating is applied to.
- `is_liked` - a boolean value indicating if the rating is positive or negative. True for positive, false for negative.
- `created_at` - timestamp detailing when the rating was created
- `updated_at` - timestamp detailing when the rating was last updated

QuestionsController

The QuestionsController class handles all of the requests that come in from users. It contains methods that cover all of the CRUD operations, including some that render forms for users to create and edit questions.

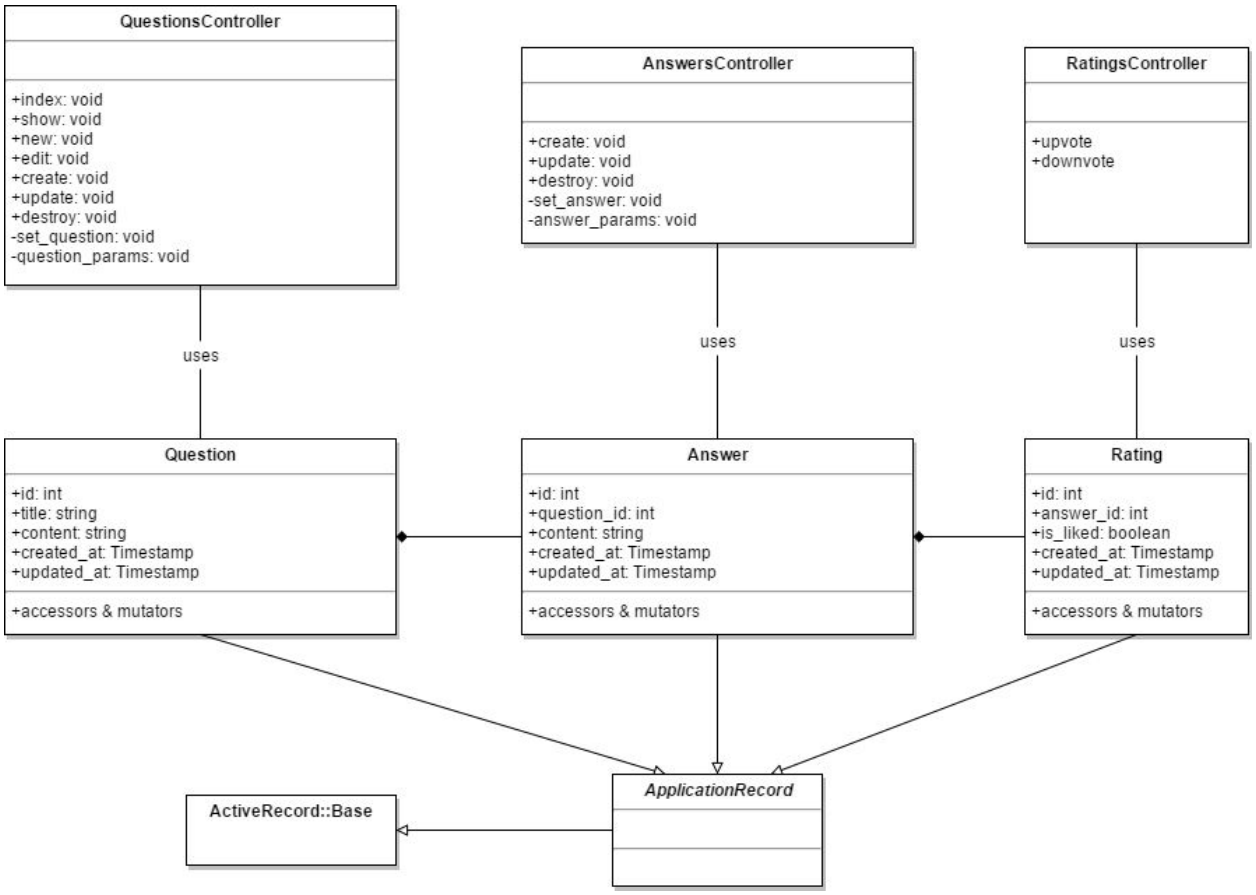
AnswersController

The AnswersController class handles a subset of the CRUD operations, because not all of them are needed for answers. The three operations the controller supports are create, update, and destroy. No other operations are needed because answers are displayed and created from within the question view, they do not need their own view.

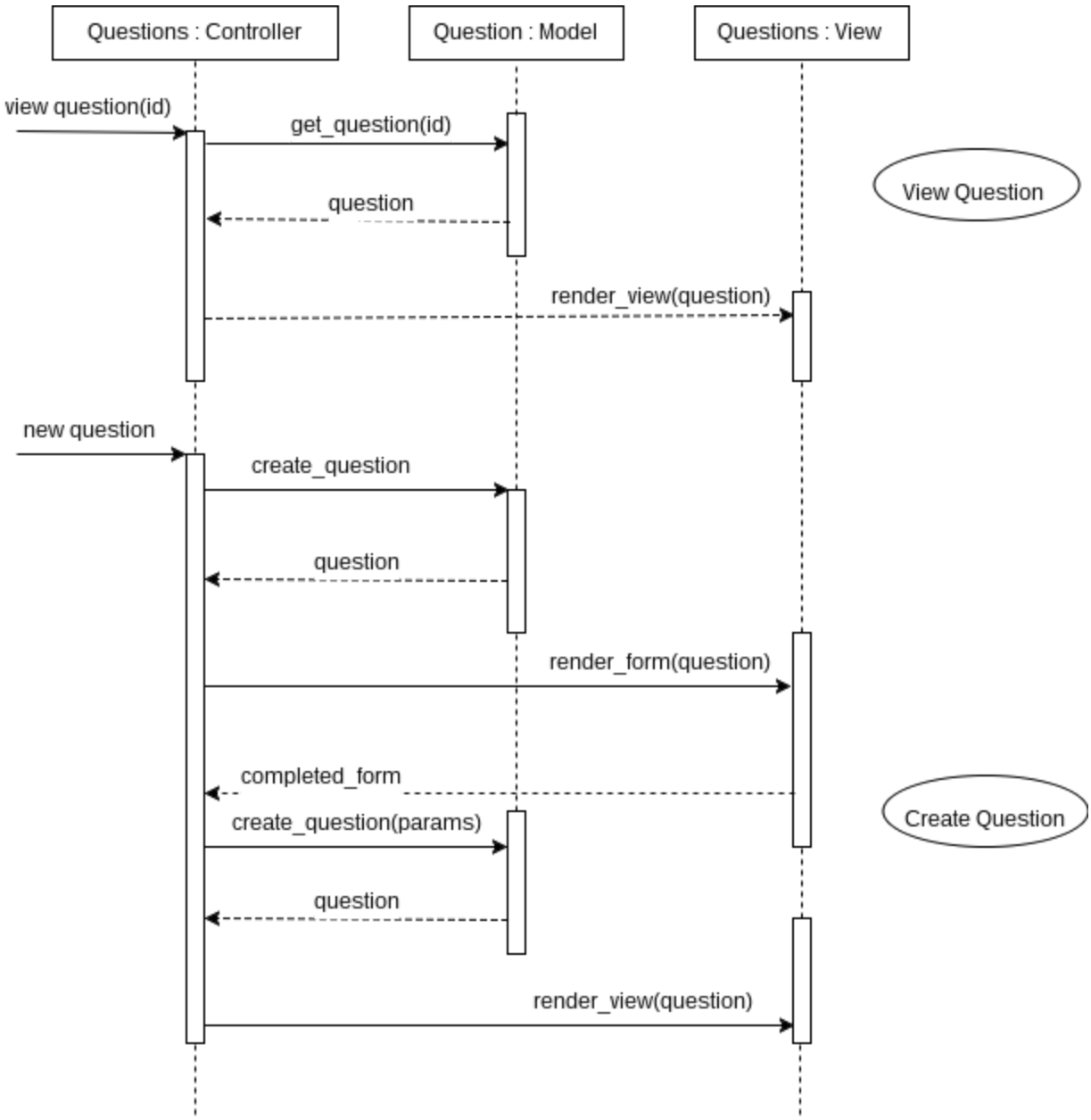
RatingsController

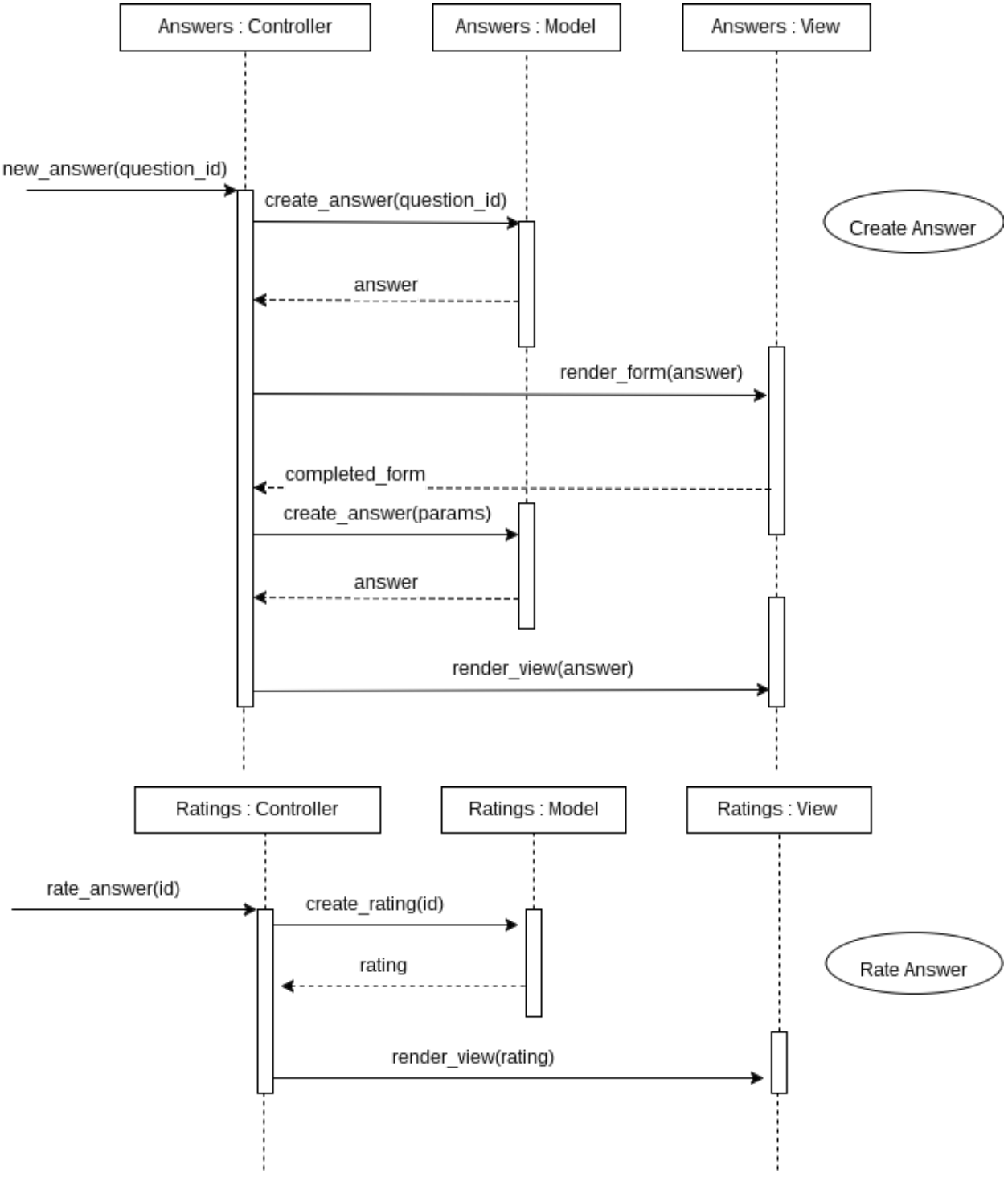
The RatingsController class, is a little bit different in that it only serves one purpose: to create ratings. As of right now, there are no users for us to easily have a way of deleting or updating a rating, so for now, they can only be created. The class has two methods, `upvote` and `downvote`, which are both aliases for a create method, the only difference being the type of rating. Calling `upvote` creates a positive rating, while `downvote` creates a negative rating.

Design Class Diagram



Interaction Sequence Diagrams




Answers : Controller
Answers : Model
Answers : View
new_answer(question_id)
create_answer(question_id)
Create Answer
answer
render_form(answer)
completed_form
create_answer(params)
answer
render_view(answer)
Ratings : Controller
Ratings : Model
Ratings : View
rate_answer(id)
create_rating(id)
rating
Rate Answer
render_view(rating)

Database Design

Initial Table Design

For the first iteration, three models are needed with the following columns:

- Question
    - id: integer
    - title: string
    - content: text
    - created_at: timestamp
    - updated_at: timestamp
- Answer
    - id: integer
    - question_id: integer
    - content: text
    - created_at: timestamp
    - updated_at: timestamp
- Rating
    - id: integer
    - answer_id: integer
    - is_liked: boolean
    - created_at: timestamp
    - updated_at: timestamp

These three tables are related in that Question has many Answers, and Answer has many Ratings. The classes outlined above are built on top of these models using the Rails built in ActiveRecord::Base class. This class provides a lot of functionality regarding querying and updating the database. For now, these tables store the bare minimum information needed to display them correctly. Eventually, they will evolve with a User table, to have references to the user that created them, along with forums that they belong to.

Rails Console Commands

- `rails g scaffold Question title:string content:text`
- `rails g model Answer content:text question:references`
- `rails g controller Answers`
- `rails g model Rating is_liked:boolean answer:references`
- `rails g controller Ratings`