УПРАВЛЕНИЕ ДОСТУПОМ В ПРИЛОЖЕНИЯХ НА БАЗЕ ФРЕЙМВОРКА RUBY ON RAILS

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Медунецкий М.А., Хоронеко М.П.

Стержанов М.В. – к.т.н., доцент

В настоящее время на многих сайтах присутствует система управления доступом и разделения прав на выполнение действий. Наибольшую популярность в веб-программировании получила система на основе ролей, которая и будет описана в этой работе.

Управление доступом на основе ролей — развитие политики избирательного управления доступом, при этом права доступа субъектов системы на объекты группируются с учётом специфики их применения, образуя роли[1]. Например, существует несколько типов пользователей сайта со своими правами. Гости могут только смотреть информацию, зарегестрированные пользователи оставлять коментарии, модераторы удалять их, а администраторы изменять содержимое веб-ресурса.

Рассмотрим пользователя сайта, который представлен моделью User. Модель пользователя должна содержать следующие поля: логин или email, пароль (обычно в целях безопасности хранится хэш пароля) и роль. Зачастую роль представлена перечислением и в базу данных сохраняется номер элемента в этом перечислении. Ruby On Rails предоставляет удобный метод взаимодействия с базой данных через класс ActiveRecord. Модель наследуется от этого класса и получает удобные методы для выполнения CRUD (создание, чтение, изменение и удаление) операций.

Для аутентификации и работы с ролями удобно использовать gem Devise [2], который содержит методы для проведения этих действий.

Аутентифика́ция — процедура проверки подлинности. Эта процедура выполняется следующим образом. Если посетитель сайта не вошёл в систему, то отправляет запрос, который содержит логин и пароль. Сервер ищет такого пользователя в базе данных и в случае успеха генерирует и возвращает токен доступа. В дальнейшем этот токен отправляется в каждом запросе, обычно заголовком. Сервер получает запрос и сопостовляет полученный токен с пользователем, которого можно получить методом current_user, если пользователь не идентифицирован, то метод возвращает nil. Devise содержит и другие методы для проверки аутентификации посетителя.

Авториза́ция — предоставление определённому лицу или группе лиц прав на выполнение определённых действий. Для установки ограничений на доступ описываются методы, которые возвращают булевое значение. Затем они прикрепляются к контроллеру, либо к его методам. Это позволяет запретить пользователям получение или изменение информации, чья группа не уполномочена на такие действия.

Таким образом, фреймворк Ruby on Rails предоставляет все необходимые инструменты для создания и поддержки приложения с управлением доступом на основе ролей, а gem Devise значительно упрощает эту работу.

Список использованных источников:

1.FerraioloD. F., KuhnD. R. (October 1992). "Role Based Access Control". 15th National Computer Security Conference: 554—563.

2. https://github.com/plataformatec/devise - Flexible authentication solution for Rails on github

РЕКУРРЕНТНАЯ НЕЙРОННАЯ СЕТЬ LSTM ДЛЯ ПРОГНОЗИРОВАНИЯ РЕЗУЛЬТАТОВ СОРЕВНОВАНИЙ ПО КИБЕРСПОРТУ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Минчук С.Г.

Жвакина А. В. – канд. техн. наук, доцент

Возрастающая популярность киберспорта, большое количество соревнований по всему миру, в том числе международных, рост призового фонда, а также признание нескольких киберспортивных дисциплин настоящим видом спорта во многих странах заставляют относиться к данным мероприятиям не как обычным развлечениям.

Прогнозирование направлено наопределение тенденций развития конкретного объекта или события на основе предыдущих данных, иными словами, анализа его состояния в прошлом и настоящем [7]. Результат игры базируется не только на данных в определенный момент, а также данных, собранных на протяжении всей игры до этого момента. Рекуррентные нейронные сети− это сети, основанные на использовании предыдущих состояний для вычисления текущего, поэтому необходимо рассмотреть возможность