

УДК 004.62

Асинхронная обработка заданий в системе поддержки разработки программного обеспечения студентами

*Сорокин Д.А., студент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»*

*Научный руководитель: Крищенко В.А., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
irudakov@bmstu.ru*

Основой образовательного процесса на кафедре «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н. Э. Баумана является разработка студентами программного обеспечения в ходе лабораторных работ, домашних заданий, курсовых проектов и выпускных работ. Для автоматизации управления этим процессом на данной кафедре была разработана система поддержки разработки программного обеспечения студентами, которая управляет репозиториями систем контроля версий и иной инфраструктурой, необходимой студенту в его работе [1].

Данная система была реализована как веб-приложение на основе библиотеки Django, но выполнение ряда операций в системе (например, создание репозитория как отдельного проекта для каждого студента потока) занимает существенное время, что затрудняет синхронное взаимодействие с её веб-интерфейсом. Кроме своей базы данных, система работает с нетранзакционными ресурсами — файлами, содержащими в себе информацию о учетных записях, репозиториях, правах доступа, и поэтому при одновременной работе нескольких пользователей с веб-интерфейсом возможна ситуация, когда производятся одновременные изменения одного и того же файла. В силу использования для ряда таких операций внешних утилит гарантировать отсутствие гонок можно только глобальной для всех потоков веб-приложения критической секцией.

В данной работе описано решение проблем параллельной работы с нетранзакционными ресурсами и асинхронного выполнения длительных операций в

кафедральной системе поддержки разработки программного обеспечения студентами. Из-за того, что содержимое некоторых файлов в системе изменяется почти всегда и критические секции для файловых операций не позволят реализовать эффективное многопоточное выполнение, было решено перевести систему на однопоточную асинхронную обработку запросов, требующих длительных операций или операций с нетранзакционными ресурсами.

Для хранения информации о заданиях, соответствующих пользовательским запросам, можно использовать очередь типа FIFO, из которой для выполнения будут выбираться задания, поступившее туда первыми. Обобщенный алгоритм обработки заданий отображен на рис.1. В ходе выполнения каждого задания система работает с собственной базой данных, а также изменяет некоторые файлы на диске. Поскольку изменение файлов ведется в один поток, то в случае неудачи при завершении транзакции в базе данных можно легко реализовать возврат файлов в исходное состояние. Таким образом, для реализации транзакционной обработки заданий нет необходимости в использовании транзакционной файловой системы. Каждое задание может закончиться успешно или с некоторой ошибкой, которую нужно сообщить пользователю.

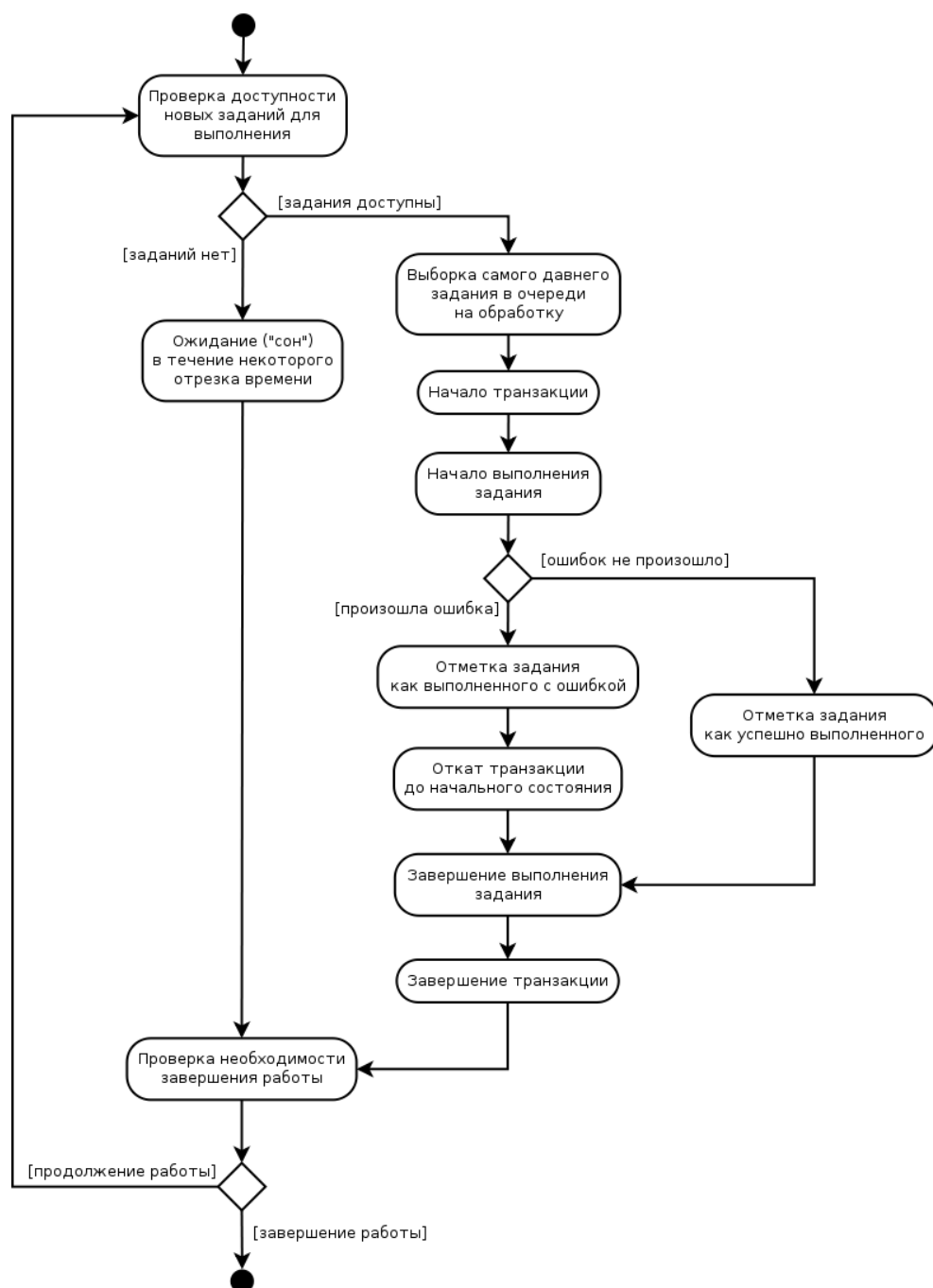


Рис. 1. UML-диаграмма активности обработчика асинхронных заданий

Для хранения очереди заданий нужно использовать некоторое постоянное хранилище, а не оперативную память. Поскольку поток заданий в данной системе не является большим, то очередь заданий размещена в базе данных системы, без использования специализированных систем обмена сообщениям на основе очередей с использованием нотификаций. Поскольку для кафедральной системы высокая производительность не является важной задачей, то после выполнения всех задач в очереди обработчик просто бездействует несколько секунд, после чего опять

опрашивает очередь заданий.

Для реализации асинхронной обработки заданий в структуру БД должны быть добавлены сущности, соответствующие заданиям, которые необходимо выполнить. Поскольку в системе было выявлено несколько существенно различающихся видов пользовательских запросов, требующих асинхронной однопоточной обработки, то была выделена сущность абстрактного задания, являющееся «базовым классом» для всех остальных конкретных заданий.

Абстрактное задание должно содержать несколько атрибутов:

- 1) состояние задания, для информирования пользователя о ходе работы над заданием;
- 2) сообщение об ошибке при выполнении задания, для информирования пользователя;
- 3) время создания задания, для их упорядочивания в виде очереди FIFO, и время завершения обработки задания.

Абстрактное задание также состоит в нескольких отношениях:

- 1) каждое задание создается по запросу некоторого пользователя через веб-интерфейс;
- 2) абстрактное задание соответствует некоторому конкретному заданию.

На рис. 2 показана ER-диаграмма сущностей, связанных с обработкой заданий. В настоящий момент в системе реализованы следующие виды заданий:

- 1) добавление, изменение и удаление отдельного проекта (репозитория системы контроля версий);
- 2) добавление, изменение и удаление роли в учебном курсе, что вызывает изменения в правах доступа ко всем проектам курса.

При обработке запроса массового создания проектов для всего учебного потока система сразу создаёт по одному заданию создания проекта на каждого студента, поскольку реализовывать атомарную массовую операцию нет необходимости.

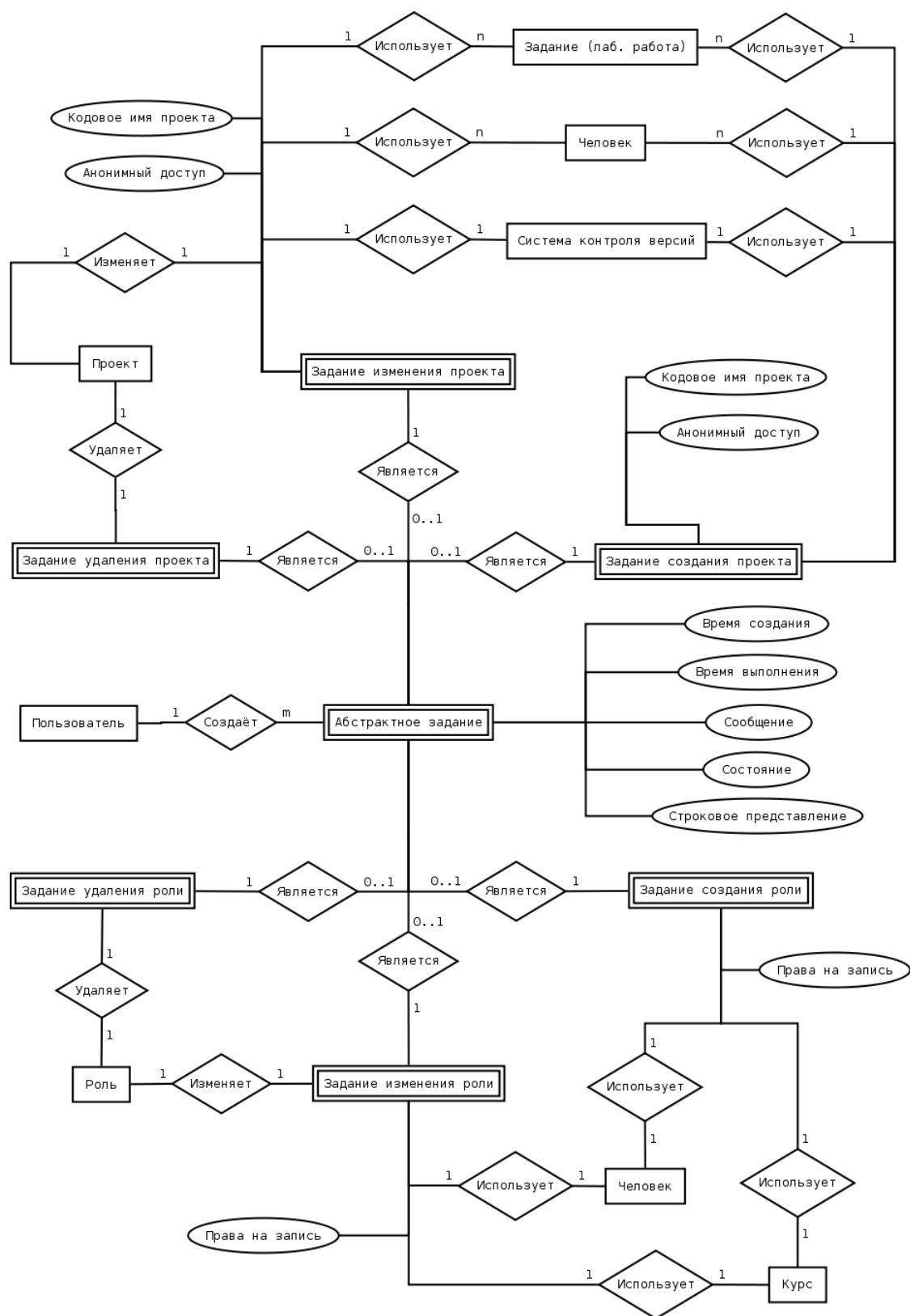


Рис. 2. ER-диаграмма сущностей заданий

Программная реализация кафедральной системы управления проектами создана на ЯП Python с использованием библиотеки Django и СУБД Postgresql. Таблицы БД, соответствующие асинхронной обработке заданий, показаны на рис. 3.

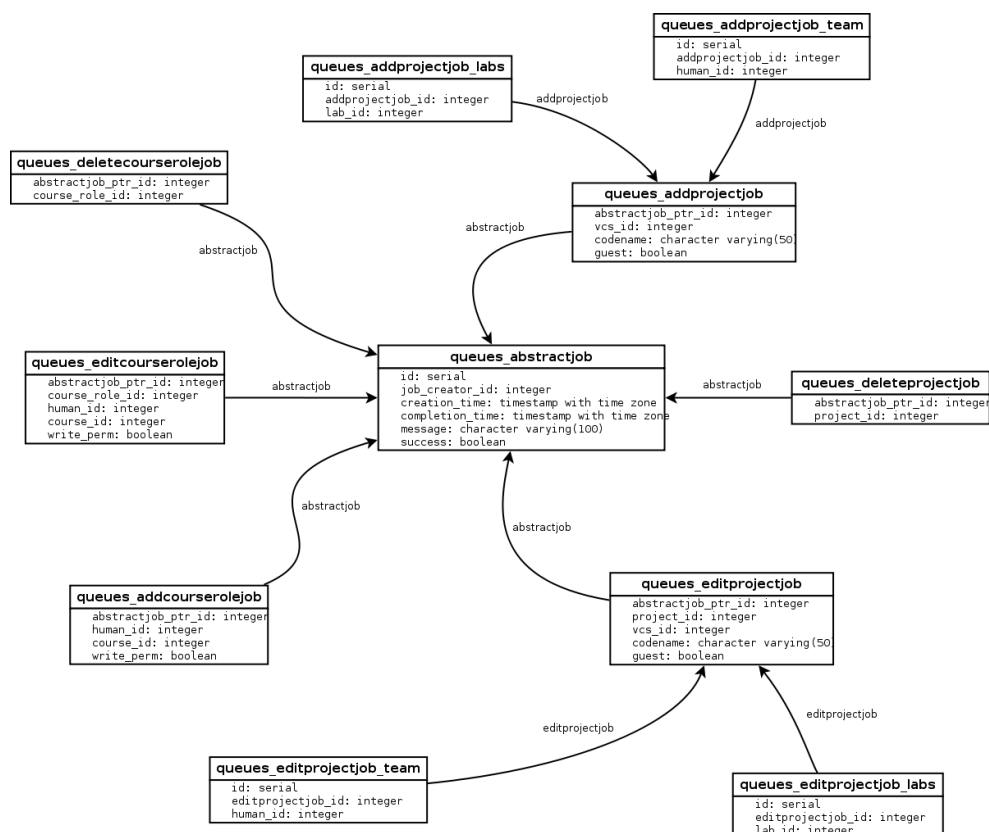


Рис. 3. Структура созданных таблиц в базе данных системы

Обработка заданий ведётся независимо от работы веб-приложения системы и связано с ним только через базу данных, хранящую задания и прочие сущности системы (рис. 4). Веб-интерфейс позволяет добавлять задания, посмотреть очередь заданий пользователя и результат выполнения заданий. Для обновления состояния заданий в веб-интерфейсе пользователя используются асинхронные HTTP-запросы. Для информирования пользователя об ошибках при выполнении его запросов используются уведомления по эл. почте.

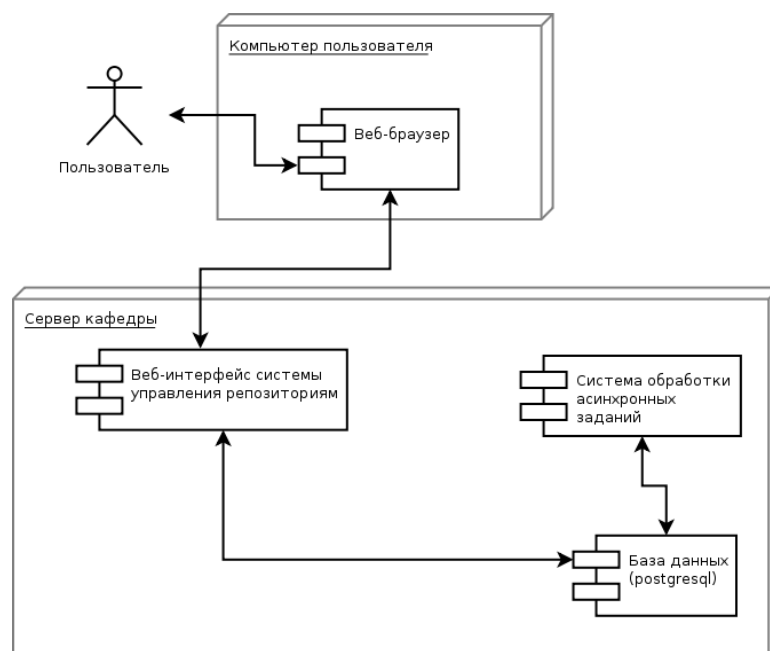


Рис. 4. UML-диаграмма развертывания системы

В рамках данной работы была рассмотрен процесс перевода кафедральной системы поддержки разработки программного обеспечения студентами на асинхронную обработку запросов по изменению репозиториях систем контроля версий. Приведены основные сущности и алгоритм обработки заданий, описаны основные моменты программной реализации. После описанных изменений пользователи системы получили возможность не дожидаться синхронного завершения длительных операций.

Список литературы

- 1) Крищенко В. А., Ковега Д. Н., Захаров Н. И. Система поддержки разработки программного обеспечения студентами // Материалы IX Всероссийской конференции «Преподавание информационных технологий в Российской Федерации». — Саратов, 2011 г. — С. 251-255.
- 2) Чан У., Биссекс П., Форсье Д. Django: Разработка веб-приложений на Python. — Пер. с англ. — СПб.: Символ-Плюс, 2010. 456с., ил. — ISBN: 978-5-93286-167-7.
- 3) Django documentation [Электронный ресурс] : официальный сайт / Lawrence Journal-World, 2003 г. — Дата обновления: 17.10.2012. — URL: <https://docs.djangoproject.com/> (дата последнего обращения: 13.12.2012).