УДК 339.138

студ. Курмашев Д.Д., Куделько В.Н.
Науч. рук. ст.преп. Блинова Е.А.
(кафедра информационных систем и технологий, БГТУ)

ORM ДЛЯ MONGODB НА NODE.JS

В современном мире ни одно приложение не обходится без базы данных. Даже небольшие блоги зачастую тесно связанны с базами данных. Рассмотри основные типы данных и попытаемся их сравнить.

Существует два основных типа баз данных: реляционная и нереляционная база данных представляет собой набор таблиц (сущностей). Таблицы состоят их колонок и строк (кортежей). Все реляционные базы данных используют «под капотом» язык SQL. SQL (StructuredQueryLanguage, язык структурированных запросов) – специализированный язык, предназначенный для написания запросов к реляционной базе данных. В нереляционных базах данных язык SQL не используется, поэтому часто нереляционные базы данных ассоциируются с NoSQL. У каждого из описанных типов баз данных есть свои достоинства и недостатки.

К основным достоинствам реляционных баз данных мы бы отнесли структурированность. Это основное отличие реляционных баз данных от NoSQL, которые позиционируют гибкость в работе с БД. Используя реляционные базы данных, вы всегда достоверно знаете с каким полями придет объект. И тут мы подошли к первому признаку в сторону выбора реляционной базы данных, наличие логических требований к данным, которые заранее определены. Но такая “строгость” к данным есть не всегда, соответственно из основного преимущества реляционных баз данных вытекает её главные главный недостаток – отсутствие гибкости, в виду чего и более низкая скорость работы, в сравнении с NoSQL.

Мир нереляционных баз данных более интересный. Идея хранения не структурированной информации с каждым днём находит все больше сторонников, и сейчас мы попытаемся объяснить почему. Основной идеей всех нереляционных баз данных является гибкость, то есть база данных не накладывает ограничений на типы данных, и на структуру хранимых документов. Соответственно уже здесь мы можем отметить первое преимущество – быстрая разработка, то есть разработчик может больше не тратить так много драгоценного времени на проектирование бд и проведение подготовительных работ. В этом заключается кардинальное различие между реляционными и нереляционными базами данных. Тем не менее стоит отметить как тако-

го противостояния между ними нет, а даже наоборот, наблюдается некоторое их сближение. Это выражается в последних заявлениях таких гигантов IT сферы как MicroSoft и Oracle, которые предлагают формы интеграции неструктурированной информации и SQL.

Существует большое количество нереляционных баз данных. Одной из самых популярных на данный момент является MongoDB. MongoDB – это нереляционная, документно-ориентированная база данных, которая использует язык программирования JavaScript в качестве языка для составления запросов. В данном случае JavaScript используется как SQL в реляционных базах данных.

Взаимодействие с базой данных может осуществляться посредствам серверного языка программирования (в основном). Самой популярной технологией для взаимодействия с базой данных MongoDB является серверный диалект языка программирования JavaScript – NodeJS. Изучив большое количество пакетов для упрощения взаимодействия работы NodeJS и MongoDB, мы пришли к выводу, что не существует полноценного средства, с помощью которого можно без потери производительности осуществлять основные базовые операции. Все пакеты, которые были нами изучены были либо слишком сложные по структуре, что приводило к существенной потери производительности, либо слишком простые, ввиду чего было слишком мало отличий от нативного драйвера. Под словом пакеты мы понимаем программное средство, которое выступает в качестве прослойки между сервером и базой данных. Чаще всего аналогичное программное средство называют ORM (ObjectRelationMapping, объектно-реляционное отображение, или преобразование) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Приняв во внимание этот факт, мы разработали программное средство, функциональность которого может быть расширен сторонними разработчиками.

Для чего оно нужно? Наше программное средство необходимо для взаимодействия с базой данных MongoDB. Перед тем как мы познакомимся со структурой нашей разработки, рассмотрим требования, которое мы поставили перед собой:

– Возможность расширения функциональности;
– минимальный размер и максимально простая реализация;
– в ходе работы с ORM не должно быть большого количества вложенных CallBack функций.

Первое требование подчеркивает смысл нашей разработки, то есть то, что наша разработка является не готовым продуктом, а прото-

типом, на основе которого можно создать более продвинутые решения. Наше разработка построена на базе нативного драйвера MongoDB для NodeJS, и не имеет никаких других зависимостей, что положительно сказывается на производительности. У стандартного драйвера, на котором построена наша разработка, есть один большой недостаток.Это старый подход к программированию на JavaScript, то есть с использованием функций обратного вызова.Учтя это, мы постарались разработать наше программное средство таким образом, чтобы у программиста, который будет его использовать не возникал так называемый “CallBackHell”, то есть ситуация большой вложенности друг в друга функций обратного вызова. Именно поэтому мы использовали ES6, в частности пару операторов async/await, которые помогает регулировать работу с асинхронным кодом.

Итоговая функциональность выглядит следующим образом:

– Insert и InsertOne;
– Find и FindOne;
– Update и UpdateOne;
– Delete и DeleteOne.

Выше представлено название основных функций, из названия понятно, зачем они предназначены. Функции без суффикса One осуществляют операции с массивами данных, а с суффиксом предназначены для выполнения операций с одни объектом. Таким образом, с помощью нашего программного средства можно выполнять CRUD (Create, Read, Update, Delete) операции.

В ходе работы приложения могут возникать ошибки базы данных, а также не исключен риск её взлома. Предусмотрев этот вариант, нами был разработан дополнительная функциональность, с помощью которого можно создавать копию базы данных, а также восстанавливать базу данных из этих копий. Эта функциональность реализован в двух функциях SaveDB и RepairFromBackup. Первая осуществляет сохранение копии базы данных в файл, в формате json. Имя файла передается в параметре, впоследствии можно настроить получение копий по расписанию, то есть вариант того что база данных будет повреждена без возможности восстановления полностью исчезает. Это также решает проблему переносимости. Если когда-нибудь появится нужда перенести базу данных на другой сервер, то можно сохранить ее в файл, а на новом сервере восстановить ее из этого файла.

В заключении стоит отметить, что хоть наша разработка и является прототипом, то есть не представляет собой законченный продукт, тем не менее она “из коробки” предоставляет хорошую функциональность, которой вполне хватит для несложных сайтов и веб-

приложений, а благодаря специально спроектированной архитектуре её легко можно будет расширить.

ЛИТЕРАТУРА

1. Репозиторий с исходным кодом проекта [Электронный ресурс] https://github.com/DenisKurmashev/mongoOrmExample/ (Дата обращения 22.03.2018)

УДК 004.42

Магистрант О. В. Луцевич
Науч.рук. доц., к.т.н. Н. Н. Пустовалова
(кафедра информационных систем и технологий, БГТУ)

SALESFORCEDX КАК НОВЫЙ ПОДХОД К РАЗРАБОТКЕ ПРИЛОЖЕНИЙ

При традиционном подходе к разработке программного обеспечения возникает одна существенная проблема: для постоянной поддержки большого проекта нужна стабильная версия программного продукта. Это необходимо для устранения обнаруженных ошибок и оперативной поставки стабильной, возможно слегка модифицированной версии заказчику, а также для разработки нового функционала в рамках проекта.

Так как иметь стабильную версию при разработке нового функционала достаточно проблематично, разработчикам приходится применять различные уловки для решения данной проблемы.

В настоящее время решить сформулированную выше проблему можно с помощью Salesforce DX.

Salesforce DX – это консольное приложение, предназначенное для поддержки совместной командной работы. Оно дает возможность создавать и индивидуально конфигурировать среды разработки. С его помощью можно под каждый новый функционал создавать полную либо частичную копию программного продукта, корректировать ее и тестировать.

В основе разработки Salesforce DX лежит понятие стартовой организации (орга) – открытого эксперимента разработчиков по созданию и управлению приложениями Salesforce на протяжении всего жизненного цикла.

Стартовая организация – это исходное и одноразовое развертывание кода Salesforce и метаданных (метаданные – это данные о данных), созданных для разработчиков и тестировщиков. Стартовая орга-