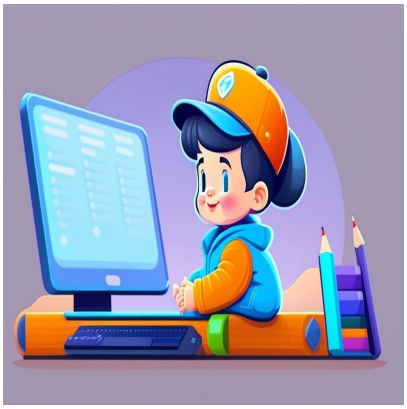


8 Платформ JavaScript ORM для эффективного кодирования

Описание

При создании полноценных приложений практически гарантированно возникает необходимость работы с базой данных. В таком приложении необходимо реализовать CRUD – возможность создания, чтения, обновления и удаления данных. Для этого необходима база данных. В том случае, если вы создаете приложение на объектно-ориентированном языке программирования, таком как JavaScript, и используете реляционную базу данных, такую как MySQL, работа с базой данных может стать сложной задачей.

Вам, как разработчику JavaScript, также придется разбираться в тонкостях работы с реляционной базой данных, выяснять синтаксис базы данных и писать сложные SQL-запросы, которые могут понадобиться вашему приложению. Помните, что реляционные базы данных хранят данные в таблицах со строками и столбцами, а JavaScript работает с объектами и связями между ними. Все это может занять много времени и сил, поэтому возникает необходимость в объектно-реляционном отображении (ORM).

Объектно-реляционное отображение (ORM)

ORM – это инструмент, позволяющий разработчикам работать с реляционными базами данных на основе объектно-ориентированных принципов. ORM является связующим звеном между кодом приложения и выбранной реляционной базой данных, позволяя разработчикам работать с реляционными базами данных, используя те же объектно-ориентированные принципы, которые они применяют в коде приложения. ORM отображают таблицы реляционных баз данных на классы, а экземпляры классов представляют собой записи или строки таблицы. Атрибуты класса используются для представления столбцов таблицы.

Это, в свою очередь, означает, что разработчики могут использовать свой язык программирования для создания, чтения, обновления, удаления, а также управления данными, хранящимися в базе данных, без необходимости написания сложных SQL-запросов. Использование ORM позволяет свести к минимуму объем SQL, который необходимо прочитать, а также избежать изучения нового языка запросов для работы с базой данных. Чтобы увидеть, как работает ORM, рассмотрим следующий запрос к MySQL для поиска пользователей из ИТ-отдела:

```
SELECT * FROM users WHERE department = 'IT';
```

Этот же запрос можно выполнить с помощью JavaScript ORM, как показано ниже. Обратите внимание на использование обычного JavaScript при выполнении того же

запроса.

```
const users = await User.findAll({
  where: {
    department: 'IT',
  },

});
```

Преимущества использования ORM

К числу преимуществ, которые разработчики JavaScript могут получить от использования ORM, относятся:

Абстрагирование сложностей базы данных

ORM маскируют сложности базовой базы данных, позволяя разработчикам взаимодействовать с базой данных с помощью языка бэкенда, а не сложного SQL. Некоторые ORM также предоставляют конструкторы запросов, которые упрощают написание сложных запросов за счет использования принципов ООП. Это позволяет разработчикам писать более чистый и удобный код, который легче отлаживать и обновлять.

Повышение производительности

ORM абстрагируют разработчиков от сложностей написания SQL-запросов и управления взаимодействием с базами данных, позволяя им сосредоточиться исключительно на бизнес-логике приложения, которая является наиболее важной его частью. Кроме того, разработчики взаимодействуют с базами данных по более привычной схеме ООП без необходимости написания большого количества шаблонного кода или выполнения повторяющихся задач. ORM также могут использоваться для автоматического посева баз данных и генерации кодов доступа к данным. Все эти факторы значительно повышают производительность труда разработчиков.

Агностичность баз данных

Ключевой особенностью ORM является то, что они позволяют писать код приложения, не зависящий от базы данных. Таким образом, код приложения не привязан к одной базе данных, а значит, можно легко менять базу данных, которую

использует приложение, без необходимости изменения значительной части кода приложения. Это очень важно, особенно когда приложение должно развиваться или поддерживать использование нескольких баз данных.

Простота управления схемами и связями

ORM упрощают процесс работы со схемами в базе данных и управление отношениями между сущностями базы данных. Некоторые ORM предлагают такие возможности, как автоматическая генерация схем на основе существующих баз данных, а большинство предоставляет методы, позволяющие легко определять и управлять отношениями между таблицами, хранящимися в базе данных.

Повышенная безопасность

ORM обеспечивают повышенную безопасность базы данных, поскольку они фильтруют данные за вас, а также используют внутренние параметризованные запросы. Параметризованные запросы – это SQL-запросы, в которых вместо непосредственного использования вводимых пользователем значений используются заполнители. Таким образом, пользовательские данные никогда не вставляются непосредственно в SQL-запрос. Это позволяет ORM защитить приложение от атак SQL-инъекций и тем самым повысить его безопасность.

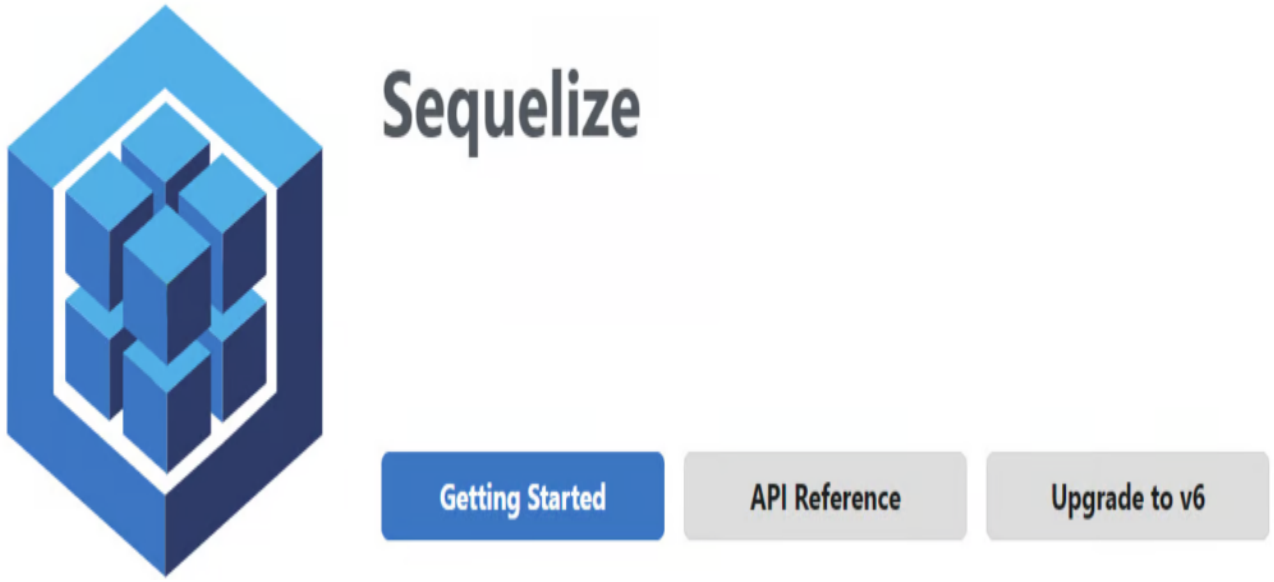
Недостатки использования ORM

Несмотря на то, что ORM имеют массу преимуществ для разработчиков, их использование сопряжено с некоторыми недостатками. Во-первых, поскольку они вводят слой абстракции поверх базы данных, это может привести к увеличению производительности и потреблению большего объема памяти. Кроме того, чтобы использовать ORM, разработчикам необходимо научиться его применять, и они не смогут использовать ORM без базового понимания SQL, чтобы знать, что на самом деле делает каждая команда. Тем не менее, ORM по-прежнему являются очень полезным инструментом для разработчиков и лучшим и самым простым способом взаимодействия с реляционными базами данных из приложений, построенных на принципах ООП. Чтобы помочь вам начать использовать ORM, вот некоторые из лучших ORM, которые вы можете использовать в своих JavaScript-приложениях.

8 Платформ JavaScript ORM для эффективного

кодирования

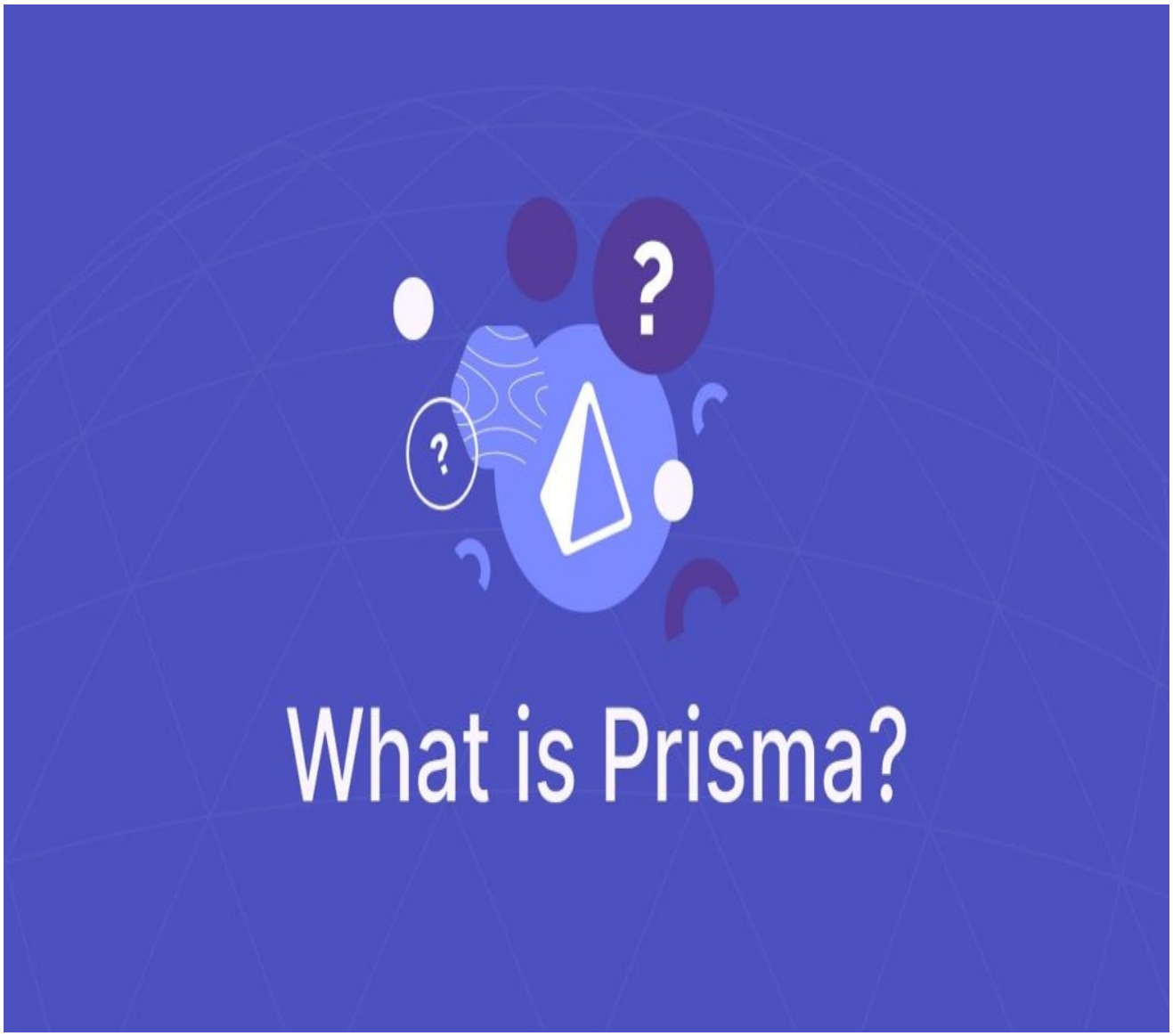
Sequelize



Согласно официальной документации, Sequelize – это современная ОРМ на TypeScript и Node.js для СУБД Oracle, PostgreSQL, MySQL, MariaDB, SQLite, Microsoft SQL Server, IBM DB2 и Snowflake. Sequelize, имеющий открытый исходный код, является очень популярным ORM для разработчиков, работающих с фреймворком Node.js совместно с реляционными базами данных. Это объясняется тем, что он обладает широким набором возможностей, которые делают работу с реляционными базами данных в Node.js простым делом. Во-первых, Sequelize – это ORM на основе обещаний, что позволяет ему поддерживать обещания, которые являются основной особенностью фреймворка Node.js.

Кроме того, Sequelize поддерживает eager loading, когда ресурсы загружаютсясразу после выполнения кода приложения, и lazy loading, когда ресурсы незагружаются сразу, пока они не понадобятся. Sequelize также имеет надежнуюподдержку транзакций, репликации чтения и проверки моделей, а также позволяет осуществлять миграцию и синхронизацию баз данных. Пользователи также могут определять ассоциации и отношения между режимами при использованииSequelize. Кроме того, Sequelize обладает богатым набором возможностей поформированию запросов, что позволяет разработчикам с легкостью создаватьсложные запросы к базе данных.

Prisma



Prisma – это ORM с открытым исходным кодом, позволяющая легко управлять базой данных и взаимодействовать с ней из любой среды JavaScript или TypeScript. Prisma поддерживает PostgreSQL, MySQL, Microsoft SQL Server, CockroachDB, SQLite и MongoDB. Кроме того, она позволяет легко интегрироваться с любым фреймворком JavaScript или TypeScript, упрощает работу с базами данных и повышает безопасность типов. Для помощи разработчикам в создании запросов в Prisma имеется функция Prisma client, которая имеет функцию автодополнения и позволяет разработчикам создавать безопасные для типов запросы, соответствующие схеме, используемой в их приложении.

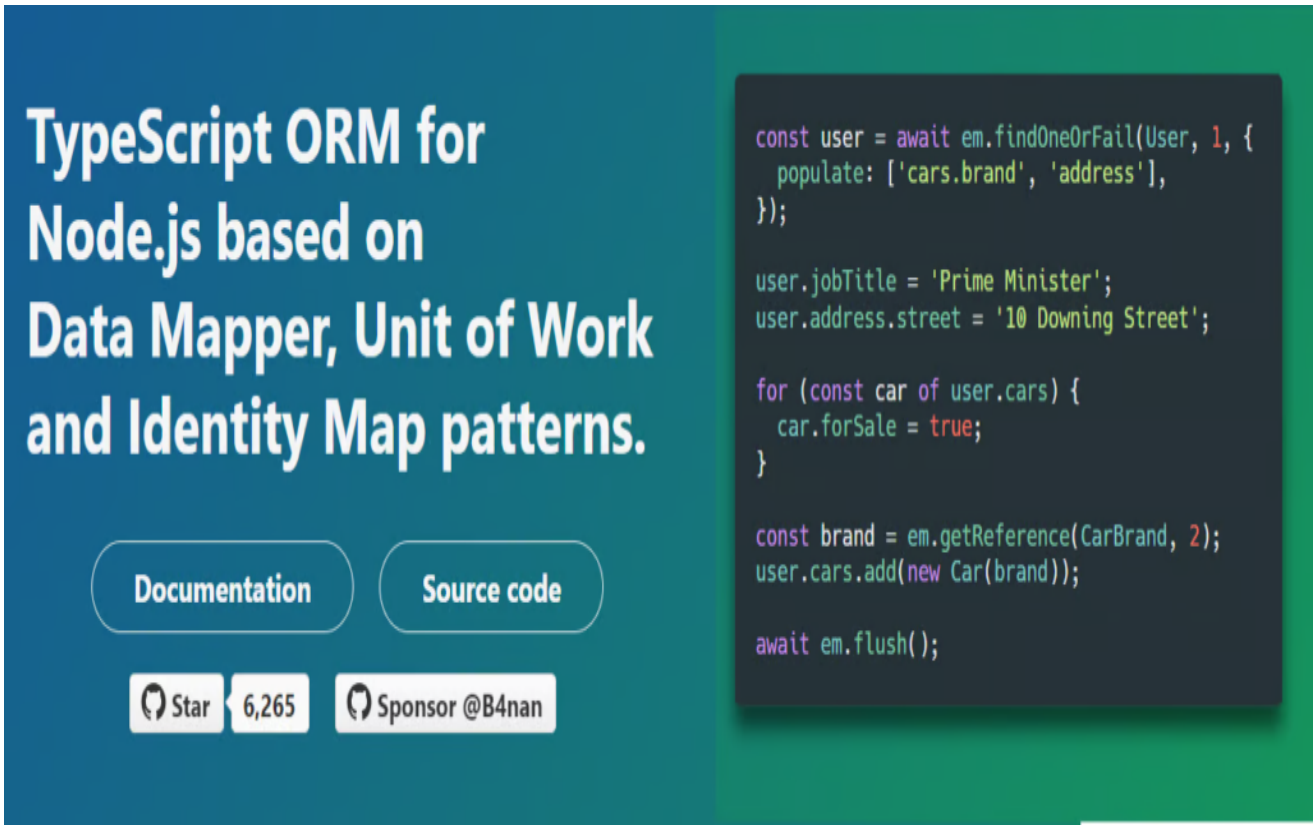
Разработчики могут создавать собственные схемы с нуля или использовать Prisma для автогенерации схем путем интроспекции существующей базы данных. Другой функцией Prisma является Prisma migrate, представляющая собой инструмент миграции схемы Prisma, который автоматически генерирует настраиваемые миграции SQL, позволяя пользователям получить полный контроль и гибкость при переносе приложений из среды разработки в среду производства. Наконец, пользователи Prisma имеют доступ к Prisma Studio, которая представляет собой пользовательский интерфейс администратора, позволяющий просматривать, исследовать, манипулировать и понимать данные, хранящиеся в базе данных. Все эти возможности делают Prisma отличным ORM для разработчиков JavaScript и TypeScript.

TypeORM

TypeORM – это ORM с открытым исходным кодом, разработанная с целью постоянной поддержки новейших возможностей JavaScript и предоставления дополнительных функций, позволяющих разработчикам создавать любые типы приложений, использующих базы данных. TypeORM поддерживает базы данных MySQL, MariaDB, PostgreSQL, CockroachDB, SQLite, Microsoft SQL Server, Oracle, SAP Hana и sql.js. TypeORM, поддерживающий языки программирования JavaScript и TypeScript, также поддерживает MongoDB, которая не является реляционной базой данных. TypeORM работает в Node.js, браузере, платформах Ionic, Cordova, React Native, NativeScript, Expo и Election.

TypeORM позволяет разработчикам работать с несколькими типами баз данных и использовать несколько экземпляров баз данных. Он также поддерживает кэширование запросов, ведение журналов, транзакции, ассоциации, отношения eager и lazy, позволяет осуществлять миграции и автоматическую генерацию миграций. TypeORM также поддерживает DataMapper, ActiveRecord, потоковые необработанные результаты, кросс-базовые и кросс-схемные запросы и предлагает пользователям мощный конструктор запросов.

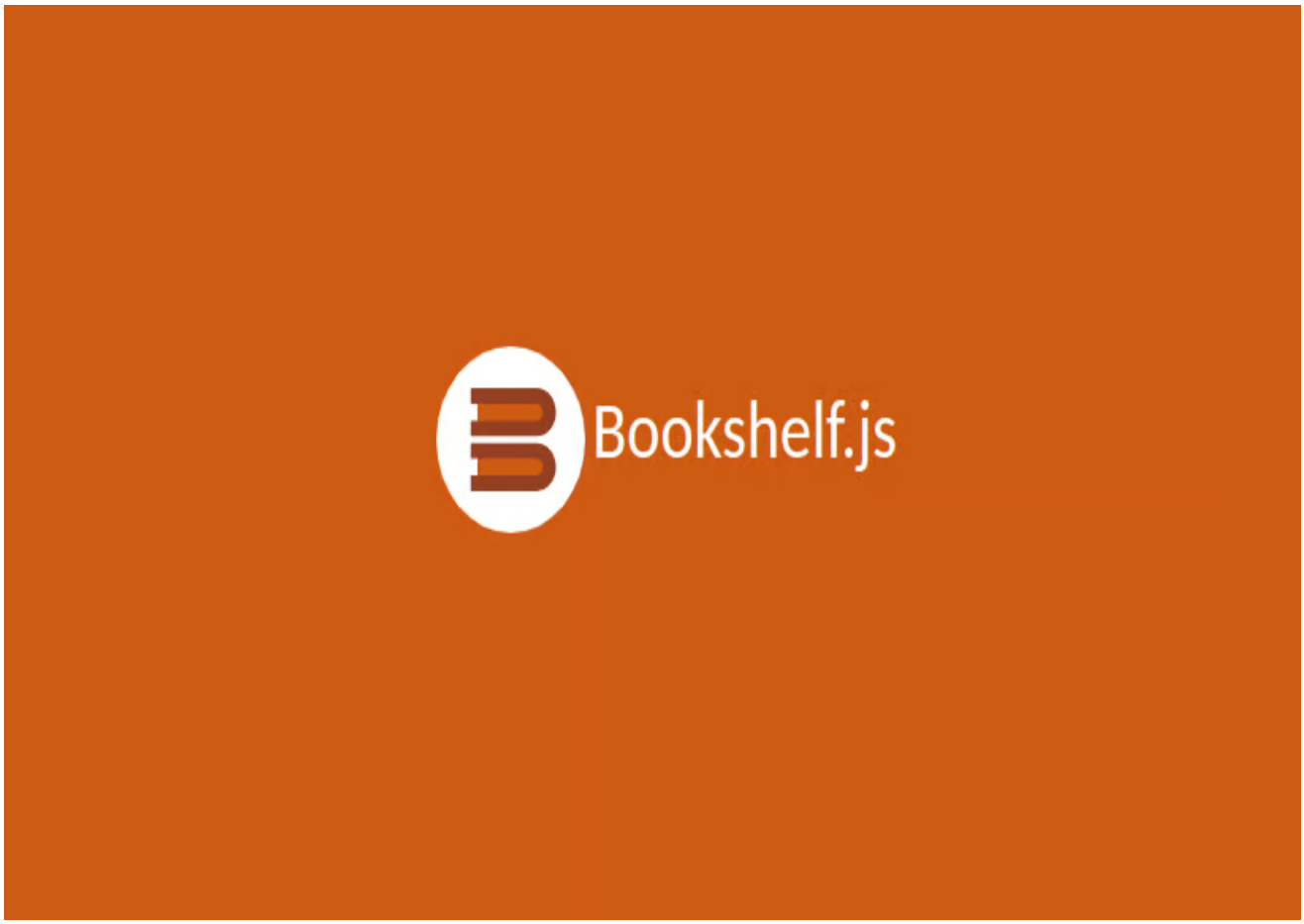
MikroORM



MikroORM – это ORM с открытым исходным кодом на языке TypeScript, поддерживающая MySQL, MariaDB, PostgreSQL, SQLite и MongoDB. В основе этого ORM лежат Datamapper, Identity Map Pattern и Unit of Work. Unit of work используется для ведения списка сущностей, затронутых бизнес-транзакцией, а также координирует запись изменений. Преимуществом является автоматическая обработка транзакций, автоматическое пакетирование всех запросов и прямая реализация логики бизнеса/домена непосредственно в используемых сущностях.

MikroORM также поставляется с метаданным QueryBuilder с поддержкой автосоединения и системой событий, которая может быть использована для подключения к жизненному циклу сущности, а также для изменения работы UnitOfWork. Засев баз данных, т.е. наполнение базы исходным набором данных, также упрощается с помощью MikroORM, поскольку в комплект поставки входит сеялка, позволяющая генерировать фальшивые данные любого объема и формы и использовать их для засева базы данных. Наконец, MikroORM также поддерживает легкую миграцию баз данных вверх и вниз.

Bookshelf.Js



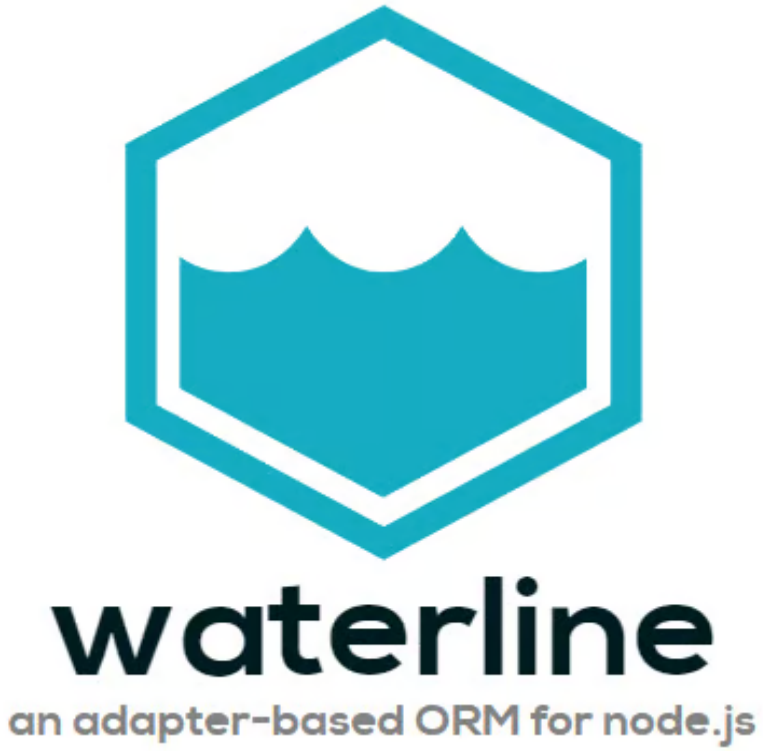
Bookshelf – это JavaScript ORM с открытым исходным кодом для Node.js. Эта ORM призвана предоставить простую библиотеку, которая может быть использована для выполнения общих задач при запросах к базам данных на JavaScript и формирования отношений между этими объектами. Bookshelf предназначена для работы с PostgreSQL, MySQL и SQLite3. Будучи Node.js ORM, Bookshelf поддерживает использование обещаний и традиционных обратных вызовов при работе с ORM из Node.js-приложения.

Кроме того, поддерживаются транзакции, полиморфные ассоциации, загрузка отношений с нетерпением/вложенным нетерпением, а также различные отношения. Несмотря на то, что Bookshelf не стоит на одном уровне с другими, более функциональными ORM, он блистает своей простотой, гибкостью и тем, что его легко читать, понимать его кодовую базу и расширять его. Если вам нужен простой и компактный ORM для ваших JavaScript-проектов, то Bookshelf – отличный выбор.

Node ORM2

Node ORM2 – это простой и легкий OPM для Node.js, поддерживающий базы данных MySQL, SQLite и Progress OpenEdge. Этот ORM позволяет легко работать с моделями в Node.js. При работе с моделями он позволяет легко создавать, синхронизировать, бросать, получать, находить, удалять, подсчитывать, а также массово создавать модели данных. Он также позволяет создавать ассоциации между моделями и определять пользовательские валидации в дополнение к встроенным валидациям, поставляемым с ним. В Node ORM2 реализовано поведение экземпляра-синглтона, которое гарантирует, что при многократном получении одной и той же строки всегда будет получен один и тот же объект, представляющий эту строку.

Waterline

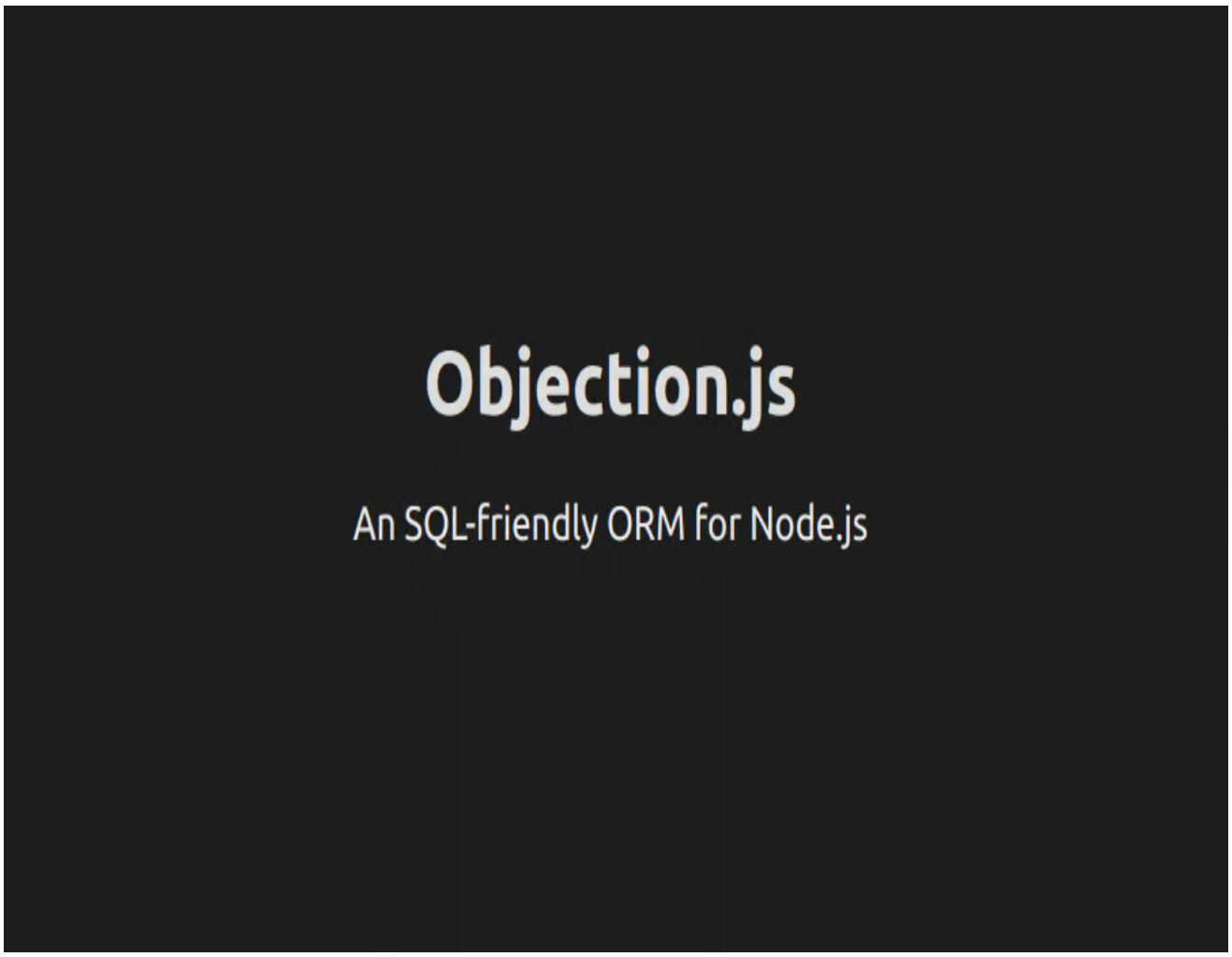


Waterline – это ORM на основе адаптеров для Node.js. Он также является ORM по умолчанию, поставляемым с фреймворком Sails. Однако Waterline можно использовать и без фреймворка Sails. Будучи ORM на основе адаптеров, Waterline поддерживает работу с несколькими системами баз данных с помощью адаптеров. Официально поддерживаются такие базы данных, как MySQL, PostgreSQL, MongoDB, Redis и локальные хранилища. Однако в Waterline также имеются адаптеры сообщества для CouchDB, SQLite, Oracle, Microsoft SQL Server, DB2, Riak, neo4j,

OrientDB, Amazon RDS, DynamoDB, Azure Table,s RethinkDB и Solr.

Waterline позволяет использовать в проекте более одной базы данных и предоставляет единый API для работы с различными базами данных и протоколами. Это означает, что код, написанный с использованием Waterline ORM, может работать с любой базой данных, поддерживаемой ORM, без необходимости изменения кода. Кроме того, при создании Waterline особое внимание уделялось модульности, тестируемости и согласованности адаптеров, что делает его очень простым в использовании и интеграции с различными базами данных.

Objection.js



Objection.js – это ORM, цель которого – не мешать вам и упростить использование всей мощи SQL и базового механизма баз данных. В этом отношении он обладает всеми преимуществами конструктора SQL-запросов и мощными возможностями для работы с отношениями. Конструктор SQL-запросов – это инструмент, который

упрощает процесс создания сложных SQL-запросов. Objection.js предлагает простой способ определения моделей и отношений между ними, а также полные возможности Create, Read, Update Delete (CRUD), использующие всю мощь SQL, в дополнение к простым в использовании транзакциям. Пользователи также могут быстро загружать, вставлять и удалять графы объектов, хранить сложные документы в виде отдельных строк и использовать проверку схемы JSON. Objection.js имеет официальную поддержку языков программирования TypeScript и JavaScript.

Заключение

Как разработчику, работающему с реляционными базами данных из приложения на JavaScript или TypeScript, лучше взаимодействовать с базой данных через ORM. Это не только упростит взаимодействие с базой данных, но и повысит производительность, сократит объем SQL, который необходимо написать, и повысит безопасность приложения. Если вы пытаетесь выбрать ORM, рассмотрите возможность использования любого из перечисленных в статье ORM, в зависимости от того, какие функции подходят для создаваемого приложения.

Дата Создания
21.07.2023