Платформа А2v10  
Разработка прикладных приложений

Оглавление

[Введение 1](#_Toc28428413)

[Платформа 2](#_Toc28428414)

[СУБД 2](#_Toc28428415)

[Прикладное приложение 3](#_Toc28428416)

[Модель данных 3](#_Toc28428417)

[Конечная точка 4](#_Toc28428418)

[Маршрутизация и конечные точки 4](#_Toc28428419)

[Первый справочник 5](#_Toc28428420)

[Второй справочник 5](#_Toc28428421)

[Документы 5](#_Toc28428422)

[Отчеты 5](#_Toc28428423)

[Архитектура платформы и прикладные приложения 5](#_Toc28428424)

[Запуск первого приложения. Где что лежит и почему 6](#_Toc28428425)

[Главное меню приложения 6](#_Toc28428426)

# Введение

Платформа **А2v10** – результат 20-летней работы на ниве создания бизнес-приложений. Это сегодняшний взгляд на то, как сегодня должна выглядеть платформа для разработки именно бизнес-приложений.

Отметим, что речь идет именно о бизнес-приложениях. Это не классический фреймворк, на котором можно писать что угодно. Это именно платформа для разработки бизнес-приложений.

Попробуем сформулировать, что же такое бизнес-приложение.

* Типовой визуальный дизайн – возможности изменения ограничены;
* Очень плотная работа с реляционными БД (именно реляционными, как бы не хотелось использовать всяческие модные POST-SQL штучки);
* Упрощение работы с бизнес-задачами – классический CRUD;
* Работа со сложными бизнес-сущностями;
* Бизнес-процессы;
* Отделение прикладной логики от платформы;
* Необходимость простой интеграции “со всем, что шевелится”;
* Возможность быстрого развертывания прикладной логики без перекомпиляции и даже без перезапуска приложения;
* Работа как в режиме web-приложения, так и в режиме настольного приложения.

# Платформа

Платформа состоит из двух частей – инструменты дизайна и исполняющая среда.

Прикладное приложение представляет собой просто набор файлов в текстовом формате. Эти файлы можно редактировать в любом текстовом редакторе. Исполняющая среда интерпретирует эти файлы “на лету”. Никакой компиляции не предусмотрено, хотя можно упаковать прикладное решение в один файл (исключительно для поддержания целостности и упрощения развертывания).

На сегодняшний день имеется два варианта исполняющей среды – набор библиотек для запуска приложения под управлением Internet information Services (IIS) либо в качестве WebApp на платформе Microsoft® Azure® и исполняющая система настольного приложения (Рис. 1).

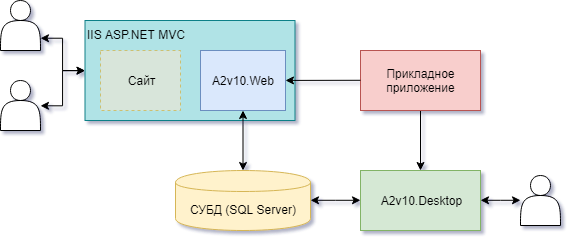


Рис. 1 Структура платформы

Отметим, что прикладное приложение остается одним и тем-же. Никаких изменений не требуется. Однако существует возможность включения/замены некоторых модулей в зависимости от исполняющей среды.

Само собой разумеется, что составной частью прикладного приложения является одна или несколько баз данных. На этом стоит остановиться подробнее.

## СУБД

Работа с СУБД в платформе **А2v10** существенно отличается от общепринятого в бизнес-приложениях.

Во многих современных системах используется так-называемые Entity Frameworks. Это механизм, который выполняет объектно-реляционное отображение (ORM – object relational mapping). Другими словами, он связывает бизнес-сущности, описанные классами на каком-то языке программирования (например С#, Java) и их представление в базе данных. Таких фреймворков существует огромное количество. Все они обладают одним очень существенным недостатком –практически не используют возможностей СУБД. А эти возможности очень велики.

Возможно появление этих фреймворков связано с попыткой “отвязать” разработчиков прикладных решений от работы с СУБД. Но проблема в том, что для разработки реальных приложений, все равно нужно очень хорошо понимать, каким образом вся эта объектная красота будет транслироваться в SQL (а СУБД ничего другого не понимает). Если этого не делать – скорость выполнения запросов будет далека от оптимальной. Это может быть не особо важным для какого-нибудь блога, но в реальных больших приложениях это может быть очень большой проблемой.

В платформе **А2v10** используется подход, позволяющий использовать все возможности СУБД по максимуму. Конечно это требует продвинутых знаний SQL, но результат того стоит.

Платформа построена таким образом, что для взаимодействия с SQL-сервером всегда (!) вызываются только хранимые процедуры. Ни при каких обстоятельствах платформа не выполняет “сырой” SQL код.

Такой подход имеет множество преимуществ:

* Независимость от структуры БД. Системе все равно, из каких таблиц получены данные. Она вообще ничего не знает о таблицах;
* Возможность использовать платформу для работы с уже существующими базами данных произвольной структуры;
* Минимизацию обращений к серверу БД, что очень важно для работы в облачном окружении;
* Возможность сохранения полных моделей произвольной сложности за одно обращение к серверу (используются табличные типы и оператор merge);
* Безопасность. Атаки типа SQL-injection невозможны в принципе. Разрешения даются только на выполнение хранимых процедур. Все классические select/insert/update просто запрещаются на уровне сервера;
* Возможность автоматической генерации текста хранимых процедур из описания модели. Поскольку основная часть работы с бизнес-сущностями шаблонная – это позволяет генерировать до 90% кода автоматически;

Кроме того, платформа позволяет работать с несколькими БД (и даже расположенными на разных серверах) в рамках одного прикладного приложения.

## Прикладное приложение

### Модель данных

Базовыми элементами, из которых строится прикладное приложение, являются *модели*. Модель это представление бизнес-сущности, такой как, например, клиент, поставщик или документ. Следует отметить, что модель это именно бизнес-сущность и она абстрагирована от таблиц БД и прочих технических подробностей.

### Конечная точка

Конечная точка описывает адресуемую сущность системы. Невозможно получить доступ к чему либо иначе как через конечную точку.

Сама конечная точка представляет собой простую папку с файлами. В папке обязательно должен находиться файл **model.json**, который и определяет поведение этой конечной точки.

Файл **model.json** описывает все возможные сценарии, которые могут обрабатываться этой конечной точкой и может включать следующие элементы:

1. Действия (**actions**) – фактически это страницы приложения, связанные с моделью. На действия могут быть назначены ссылки в меню, они имеют свой собственный адрес (URL), который можно сохранить в закладках браузера или переслать по почте. Следует обратить внимание, что при переходах от действия к действию теряется текущий контекст, так как фактически браузер открывает новую страницу.
2. Диалоги (**dialogs**) – вспомогательные модальные диалоги. Отображаются поверх текущей страницы без потери контекста. Могут иметь доступ к вызывающему контексту. Обычно используются для выбора элемента, редактирования какой-то сущности и т.д.
3. Всплывающие окна (**popups**) – вспомогательные немодальные окна. Автоматически закрываются при потере фокуса. Обычно используются для показа дополнительной информации.
4. Отчеты (**reports**) – действия, которые имеют представление, отличное от стандартного html. Это могут быть отчеты из внешних систем, xml-файлы и т.д.
5. Команды (**commands**) – действия, которые оперируют моделью, и не имеют внешнего представления. Это могут быть вызовы хранимых процедур SQL, вызов методов внешних СLR-библиотек, обращения к внешним сервисам и прочее.
6. Файлы (files) – действия, связанные с загрузкой/выгрузкой и обработкой внешних файлов, таких как изображения, текстовые документы и прочее. Файлы чаще всего рассматриваются системой как двоичные объекты (хотя это и зависит от того, как они хранятся в базе данных). Другими словами, файлы это то, что обычно загружается/выгружается на сайтах интернета.

 Кроме файла **model.json**, в папке обычно находятся файлы шаблонов, скриптов, представлений, отчетов и т. д.

Файловая система

16:36

Конечная точка redirect.json – структура, маршруты и действия

# Маршрутизация и конечные точки

Структура URL

# Первый справочник

* Справочник контрагентов
* Таблица элементов
* Возможные действия
* Модель индекса. Процедура .Index
* Модель элемента. Процедуры .Load, .Metadata, .Update
* Отображение индекса. Вызов диалогов создания/редактирования элемента
* Диалоги редактирования. Представление. Шаблоны, Валидаторы.
* Удаление элементов

# Второй справочник

* Справочник товаров
* Связанные таблицы (единицы измерения)
* Диалоги выбора (browse)

# Документы

* Таблицы документов и строк
* Модель документа. Связанные таблицы в модели данных

# Отчеты

* Простой отчет
* Отчет с группировками
* Перекрестный отчет

# Архитектура платформы и прикладные приложения

* Состав платформы
* Понятие прикладного приложения
* Элементы приложения
* Модель данных – основа приложения
* Маршрутизация
* Обработка модели на сервере
  + Хранимые процедуры
  + Метаданные
  + Формат возвращаемых данных
* Работа с данными на клиенте
* Представления
  + Xaml и html
  + Связывание
  + Элементы управления
  + Контейнеры
  + Компоненты
* Контроллер и данные
  + Модель на клиенте
  + Методы контроллера
* Шаблоны
  + Свойства
  + Валидаторы
  + Команды
  + События
  + Делегаты

# Запуск первого приложения. Где что лежит и почему

* SQL-Server
* IIS
* Пул приложений
* Доступ к файловой системе
* Web.config
* Connection Strings
* Как все это работает вместе

# Главное меню приложения

* Таблица меню приложения
* Списки контроля доступа и механизм назначения прав