Лекция

ПИ

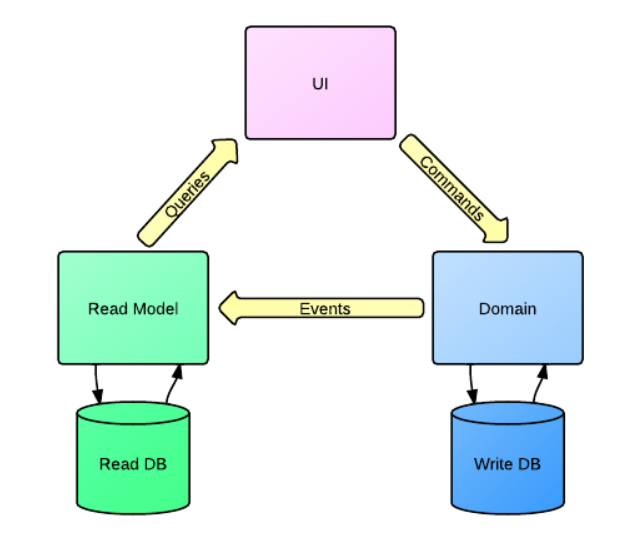
ПОИТ-4

**Domain-Driven Design (DDD)**

1. **DDD:** Domain-Driven Design - предметно(проблемно)-ориентированное проектирование: набор принципов направленных на создание систем объектов со сложной предметной областью.
2. **DDD:** цель–ускоритьпроцесс проектирование программного обеспечения.
3. **DDD:** введен Э. Эвансом в книге «Domain-Driven Design», 2004г.
4. **DDD:** Domain – предметная область. **Эксперт домена** – специалист в предметной области.
5. **DDD:** 1) создание единого языка домена (Ubiquitous Language).Язык - набор однозначно определенных терминов. Тезаурус.
6. **DDD:** 2) выделение ограниченных контекстов (имена).
7. **DDD:** 3) определение контрактов (имена) (интерфейсов взаимодействия контекстов); контракты описывают команды (приводит к изменению состояния контекста, могут быть отклонены) и события (сообщение системы об изменении состояния, на них могут подписываться другие контексты, могут быть проигнорированы, но не могут быть отклонены).
8. **DDD:** 4) описание контекстов.
9. **DDD:** система имеет 4уровня: interface (пользовательское взаимодействие, взаимодействие с другими системами), application (реализация, технические аспекты приложения, хранение данных, создание данных), business (реализация доменного языка), infrastructure (платформа и дополнительное программное обеспечение).
10. **DDD:** рекомендует паттерны: Value Object, Entity Object, Aggregate (граница транзакции, граница Entity), Repository, Factory, Service.
11. **DDD:** модули (высокая степень независимости).
12. **DDD:** документация на языке
13. **DDD:** используется для больших проектов, длительностью более 6 месяцев.

**Command-Query responsibility segregation(CQRS)**

1. **CQRS**: разделение ответственности на команды и запросы ; CQS – command-query separation(Б. Майер) = принцип императивного программирования (команда-память-команда-память, ассемблер – типичный императивный язык). Основной принцип – команда-чтение не вызывает побочного эффекта(не изменяет состояние объекта).
2. **CQRS**: контрактное программирование (Б.Майер): спецификация методов (интерфейсы), предусловие, постусловие, инвариант (обязательные соглашения в методе), исключения.
3. **CQRS**: перенос основных принципов CQS на архитектуру (М.Фаулер)



1. **CQRS:** обычно 2 базы данных: для записи и для чтения. Базы данных синхронизируются.

**Event Sourcing (ES)**

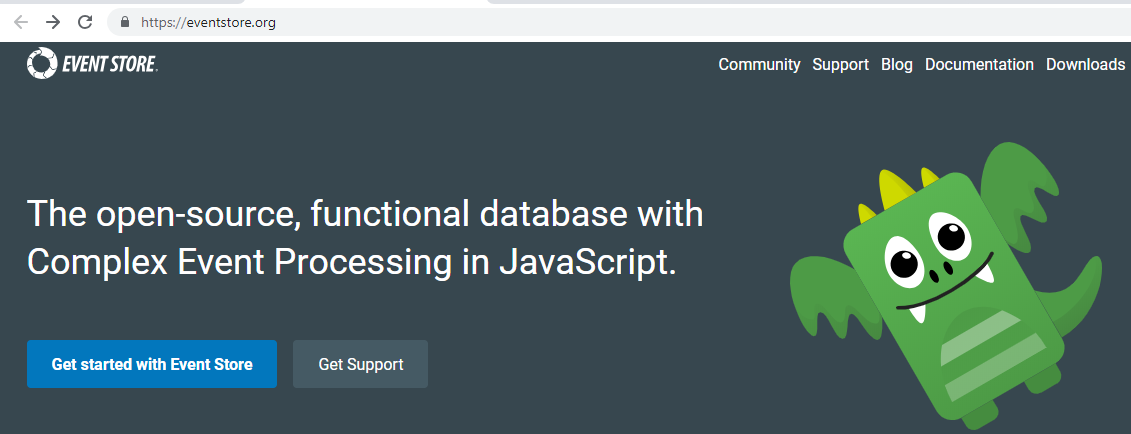
1. **ES:** подход к хранению данных, при котором вместо конечного результата хранится череда событий произошедших с некоторой сущностью.
2. **ES:** каждое событие имеет имя.
3. **ES:** произошедшие события неизменны.
4. **ES:** проекция - вычисленные результаты для UI (аналог view).
5. **ES:** ориентация на процесс (workflow driven), а не на данные (data-input based).
6. **ES:** обычно применяется в архитектуре CQRS для реализации WriteDB.

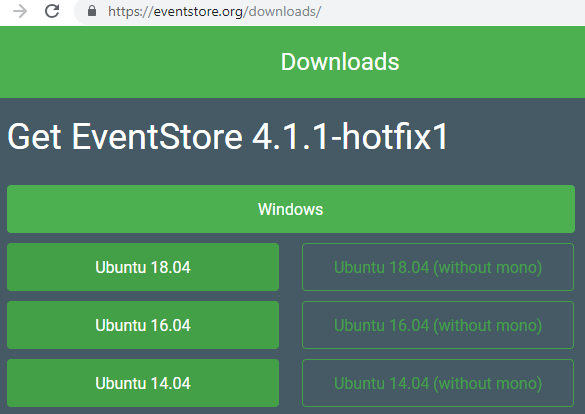


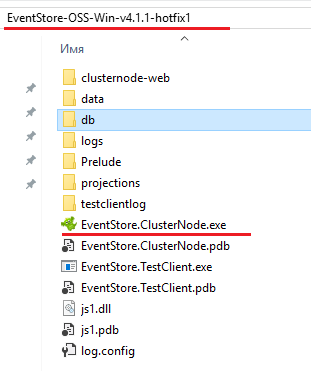
1. **ES:** реализация

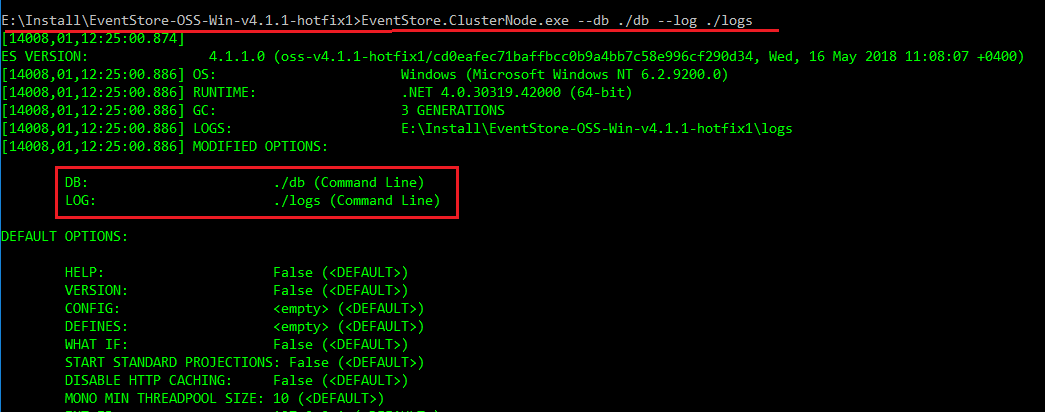
**EventStrore**

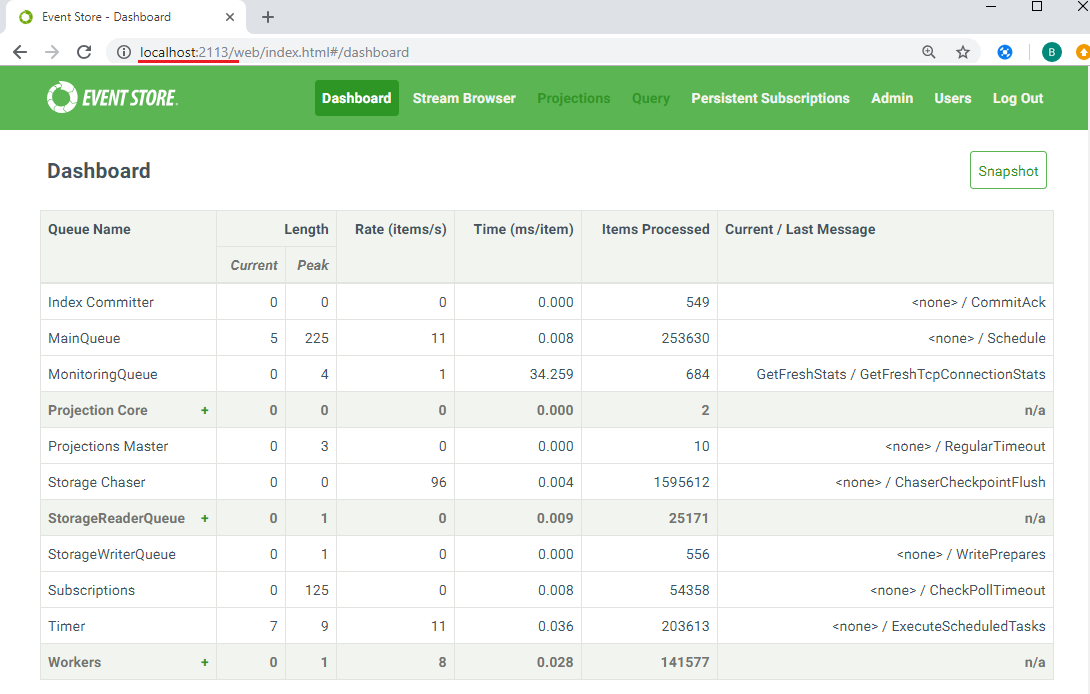
1. **EventStore:** функциональная база данных с комплексной обработкой событий в JavaScript.

****

****

****

****

****

