Отчет за период 17.04-26.04
Запланировано задач к текущему дню: 7
Выполнено: 7
Всего выполнено: 36/36.
Общий процент выполнения плана: 100%
Остается недель до окончания: 0

Прочтение книги завершено. В целом доволен чтением, получил первый взгляд на устройство распределенных проблем, какие проблемы там решаются. Какие-то схемы можно применить и при построении дизайн системы в мобильном приложении.

День 18.04
Продолжаю рассмотрение паттернов репликации

The active-active replication pattern
Держите множество копий запущенного сервиса в разных локациях, и выполняйте все изменения на всех одновременно

День 19.04 - 23.04
Приступил к главе “Управления ресурсами”.

Шаблон “Инкапсуляция ресурсов”. Ресурсы и их жизненный цикл обязаны принадлежать одному компоненту

Шаблон “Кредит ресурсов”. Предоставьте Клиенту эксклюзивный временный доступ к ограниченным ресурсам без передачи прав владельца.

Шаблон “Сложная команда”. Отправляйте составные инструкции ресурсу, чтобы избежать чрезмерного исплользования сети.

Шаблон “Пул ресурсов”. Владелец управляет пулом ресурсов.

Заключение по главе. Она посвящена моделированию и управлению ресурсами системы, были рассмотрены паттерны, соответствующие реактивной системе и способствующие решению различных сценариев. В целом пока информация для меня нерелевантна для немедленного применения, но получить первый обзор было полезно.

Приступил к чтению главы “Message Flow patterns”.
Паттерны:
- Request-Response. Включите в сообщение адрес, на который вернется ответ.

- Self-contained Message. Каждое сообщение содержит всю необходимую для выполнения запроса информацию, как и для получения ответа
- The Ask pattern. Делегируйте обработку ответа выделенному компоненту
- Forward Flow pattern. Там, где возможно, позвольте информации и сообщениям поступать непосредственно к месту назначения
- The Aggregator Pattern. Создайте “временный” (эфемерный) компонент, если для вычисления результата вызова сервиса требуется получить ответ от нескольких других сервисов
- The Saga pattern. Создайте временный (эфемерный) компонент для управления исполняющимися экшенами, распределенных по нескольким компонентам.
- The business handshake pattern. Включите в сообщение идентифицирующие и/или упорядоченную информацию и повторяйте запроса, пока не получите подтверждение. Вкратце, паттерн про то, чтобы исполнять запрос только единожды (без дублей). Так как в нестабильных сетях может произойти что угодно, из-за чего запросы будут отправляться повторно.

Заключение по главе. Ознакомился с механизмами для моделирования информационных потоков в реактивных системах. Рассмотрели схему запроса-ответа посредством полных и автономных сообщений. Для сложных взаимосвязей между компонентами рассмотрены шаблоны Aggregator и Saga. Добавили надежности, используя шаблон business handshake.

День 24.04

Перешел к главе “Flow control patterns”.

- The Pull pattern. Попросите потребителя запросить у продюсера пакет данных
- The Managed Queue pattern. Управление очередью сообщений
- The Drop pattern. Дропнуть запрос предпочтительнее бесконтрольного сбоя в системе
- The Throttling pattern. Учитывайте производительность в соответствии с контрактом других сервисов.

Узнал о rate-limiting алгоритме Token bucket.

Заключение по главе. Рассмотрели техники управления скоростью взаимодействия. The Pull pattern приводит в соответствие скорость обработки источника данных и потребителя, задавай нижний темп. The Managed Queue регулирует входящие и исходящие сообщения с помощью очереди и делает разницу в скорости между consumer и producer измеримой. The Drop pattern обеспечивает эскалацию для шаблона manager queue, когда разница в скорости обработки слишком большая, чтобы ценой понижения функциональности сервиса можно было обработать запрос. И шаблон The Throttling регулирует скорость поступления сообщений в соответствии с параметрами конфигурации

День 25.04

Перешел к главе “State management and persistence patterns”.

Рассмотрим следующие шаблоны:

- The Domain Object
- The Sharding
- The Event-Sourcing
- The Event Stream

Domain Object. Разделяйте бизнес-логику от управления состоянием и коммуникаций

Sharding. Масштабируйте управление большим количеством доменных объектов, группируя их в шарды, основанные на уникальных и постоянных свойствах объекта.
The Event-Sourcing. Выполняйте изменения состояния только путем применения событий. И делайте их долговечными, сохраняя события в журнал. **https://martinfowler.com/eaaDev/EventSourcing.html**

The Event-Stream. Публикуйте события, генерируемые компонентом, чтобы остальные части системы могли извлекать из них информацию.

Заключение по главе. В этой части рассмотрели шаблоны по управлению состоянием в реактивной системе и как их использовать вместе. Domain object отделяет бизнес-логику от выполнения посредством сообщений и позволяет тестировать и описывать логику, не заботясь о проблемах распределенной системы или асинхронности. Sharding позволяет эффективно хранить произвольное количество объектов при наличии кластера с достаточными ресурсами. Event-Sourcing превращает деструктивное обновления состояний в накопление информации, представляя, что полная история состояния объекта представлена записями в логе (журнале), которые он генерирует. Event-Stream использует сохраняемые события изменения для реализации надежного и масштабируемого распространения информации по всей системе. Отправленные события могут быть использованы любым количеством заинтересованных клиентов для получения новой информации.