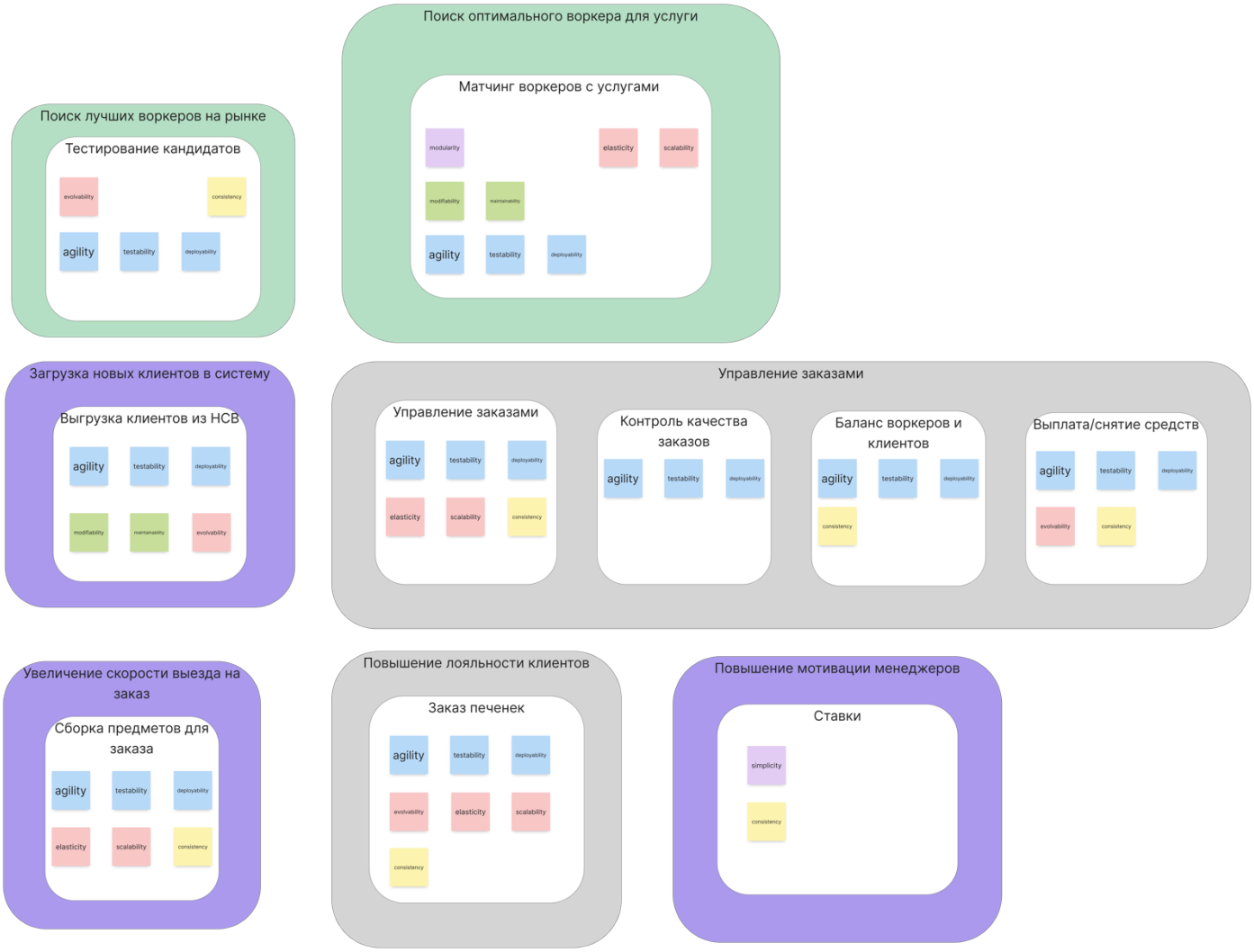
ADR-001: Выбор архитектурного стиля для Make Cats Free

Status: *Accepted*

Context

Основываясь на требованиях и консёрнах стейкхолдеров, было определено, что необходимо реализовать систему из десяти боундед-контекстов, каждый из которых имеет свой набор характеристик.



Набор характеристик для каждого найденного элемента Посмотреть изображение поближе

Каждая из характеристик была найдена по своим причинам:

- понятные значения для *scalability* и *elasticity*, связанные с поддоменами «управления заказами», «сборка предметов для заказа», «заказ печенек» и «матчинг воркеров с услугами». Это характеристики из требований и новой информацией от менеджеров о том, что наш проект понравился котам из НСВ;
- *modifiability* и *maintainability*, которые относятся к двум поддоменам «тестирование кандидатов» и «матчинг воркеров с услугами». Взял из характеристик core-поддоменов. Важно учитывать, что для этих core-поддоменов нужен низкий каплинг;
- *evolvability*, относящийся к четырем боундед-контекстам поддомена «тестирование кандидатов», «выгрузка клиентов из НСВ», «сборка предметов для заказа», «заказ печенек» и «выплата/снятие средств»;
- *scalability*, *modifiability*, *maintainability*, *testability*, *agility* и *evolvability* для всего, потому что система растёт;

- *modularity* для боундед-контекста «матчинг воркеров с услугами», которое исходит из консёрнов разработчиков (дата сайентистов);
- *consistency* для данных в боундед-контекстах «тестирование кандидатов», «управление заказами», «сборка предметов для заказа», «заказа печенек», «баланс воркеров и клиентов» и «выплата/снятие средств» из-за CatFinCompliance.

Кроме характеристик, в системе есть следующие ограничения:

- соблюдение CatFinCompliance, который говорит об особом способе хранения данных и особой наблюдаемости за системой. Компания не хочет повторять опыт с маски-шоу, которые были в Happy Cat Box.

Хотя каждый из элементов содержит свой набор характеристик, необходимых только ему, есть три характеристики, которые являются общими для всей системы: *agility* (низкий TTM), *deployability* (критично проверять новые гипотезы с максимальной скоростью) и *testability* (критично проверять новые гипотезы с максимальной надежностью). Для этих трёх характеристик и будет выбираться архитектурный стиль всей системы.

Для этого будет рассмотрено семь архитектурных стилей

	layered	modular monolith	service-based	microservices	microkernel	pipline	event-driven
agility	★	★★	★★★★	★★★★★	★★★	★★	★★★
abstraction	★	★	★	★	★★★	★★★	★★★★
configurability	★	★	★★	★★★	★★★★	★★	★★
cost	★★★★★	★★★★★	★★★★	★	★★★★★	★★★★★	★★★
deployability	★	★★	★★★★	★★★★	★★★	★★	★★★
domain part.	★	★★★★★	★★★★★	★★★★★	★★★★★	★	★★★★★
elasticity	★	★	★★	★★★★★	★	★	★★★★
evolvability	★	★	★★★	★★★★★	★★★	★★★	★★★★★
fault-tolerance	★	★	★★★★	★★★★	★	★	★★★★★
interation	★	★	★★	★★★	★★★	★★★	★★★
interoperability	★	★	★★	★★★	★★★	★★	★★★
maintainability	★★	★★★	★★	★★★★★	★★★★	★★★	★★★
modifiability	★★	★★★	★★	★★★★★	★★★★	★★	★★★
modularity	★	★	★★★★	★★★★★	★★★	★★★	★★★★
performance	★★★	★★★	★★★	★★	★★★	★★	★★★★★
reliability	★★★	★★★	★★★★	★★★★	★★★	★★★	★★★
scalability	★	★	★★★	★★★★★	★	★	★★★★★
security	★★	★★	★★★	★★★★★	★★★	★★★	★★★★
simplicity	★★★★★	★★★★★	★★★	★	★★★★	★★★★★	★
testability	★★	★★	★★★★	★★★★	★★★	★★★	★★
workflow	★	★	★	★	★★	★★★★	★★★★★

Все стили, которые будут рассматриваться для принятия решений

Decision: микросервисный стиль

Для системы были выбраны три основные характеристики: *agility*, *deployability* и *testability*. По общей таблице со стилями видно, что есть ровно два варианта, которые подойдут: service-based или микросервисы.

	layered	modular monolith	service-based	microservices	microkernel	pipline	event-driven
agility	★	★★	★★★★	★★★★★	★★★	★★	★★★
abstraction	★	★	★	★	★★★	★★★	★★★★
configurability	★	★	★★	★★★	★★★★	★★	★★
cost	★★★★★	★★★★★	★★★★	★	★★★★★	★★★★★	★★★
deployability	★	★★	★★★★	★★★★	★★★	★★	★★★
domain part.	★	★★★★★	★★★★★	★★★★★	★★★★★	★	★★★★★
elasticity	★	★	★★	★★★★★	★	★	★★★★
evolvability	★	★	★★★	★★★★★	★★★	★★★	★★★★★
fault-tolerance	★	★	★★★★	★★★★	★	★	★★★★★
interation	★	★	★★	★★★	★★★	★★★	★★★
interoperability	★	★	★★	★★★	★★★	★★	★★★
maintainability	★★	★★★	★★	★★★★★	★★★★	★★★	★★★
modifiability	★★	★★★	★★	★★★★★	★★★★	★★	★★★
modularity	★	★	★★★★	★★★★★	★★★	★★★	★★★★
performance	★★★	★★★	★★★	★★	★★★	★★	★★★★★
reliability	★★★	★★★	★★★★	★★★★	★★★	★★★	★★★
scalability	★	★	★★★	★★★★★	★	★	★★★★★
security	★★	★★	★★★	★★★★★	★★★	★★★	★★★★
simplicity	★★★★★	★★★★★	★★★	★	★★★★	★★★★★	★
testability	★★	★★	★★★★	★★★★	★★★	★★★	★★
workflow	★	★	★	★	★★	★★★★	★★★★★

Основываясь на *agility*, *deployability* и *testability*, у нас есть только два стиля, которые могут подойти: микросервисы или service-based

Чтобы выбрать единственный стиль из двух, необходимо решить, будет использоваться одна общая база на все сервисы или каждый из сервисов будет иметь свою базу данных.

Для этого стоит учитывать следующие детали:

- сервис биллинг должен соблюдать комплаенс, поэтому база должна быть отдельной. Из-за требований к *consistency* для него подойдёт реляционная изолированная база;
- опираясь на консёрны разработчиков (дата сайентистов), для них подойдёт графовая база данных;
- для сервиса отбора кандидатов важна *consistency* + легкое масштабирование и обработка клиентских данных поэтому им подойдёт реляционная изолированная база;
- остаётся четыре сервиса для определения: управление заказами, загрузки новых клиентов, ставок и сборки. Во всех случаях нет ограничений для выбора типа базы, но остаётся выбор: использовать одну базу, чтобы оставлять service-based-стиль, или разделять данные на четыре. В этом случае все базы имеют пересечение по данным в области заказов и клиентов, а в остальном отличаются. Минус в том, что если использовать одну базу на четыре сервиса, то появятся проблемы с совместной работой над схемой данных, а плюсов нет.

Основываясь на деталях, связанных с базой данных и архитектурным стилем, решено делать отдельную базу под каждый сервис. А следовательно, и выбрать микросервисный стиль для Make Cats Free.

Compliance

Так как нет вариантов автоматической проверки реализации архитектурного стиля, то вся проверка будет делаться вручную посредством изучения проектной документации, кода и автоматически генерируемой карты сервисов, основанной на инструментах трассировки.