Ревью индивидуального PR – Максим Брыков, 2 итерация

Компоненты	Базовый вариант	Великолепный вариант	Комментарий
1. Вёрстка. Auto Layout	✅Добавлены все констрейнты, чтобы все элементы располагались согласно макетам и корректно масштабировались на разных телефонах (на всех) без потери критической информации вроде цены и названия; ✅Текст должен быть читаемым, без лишних констрейнтов, без конфликтов. Допущения: - Максимум один конфликт констрейнтов или приоритетов; - 2-3 отступа слегка отличаются от макета (например, если расположение на 10 пикселей отличается от макета)		Отлично!
2A. Вёрстка в коде	✅Реализована вёрстка экранов в коде без использования Interface Builder ✅Использованы best-practices для реализации вёрстки, такие как Auto Layout и Stack Views. Допущение: Можно использовать SnapKit или другие сторонние библиотеки для упрощения вёрстки	Реализованы множественные переиспользуемые UI компоненты там, где это возможно – 🍏	Хороший уровень верстки кодом. Не хватает насмотренности в том, как организовывать верстку внутри класса: рекомендация не использовать создание вьюшки с кложурой внутри функций – слишком громоздко. Есть переиспользуемая вьюшка AvatarView
2B. Вёрстка в сторибордах	-		-
3. Вёрстка таблицы	✅Корректно свёрстана ячейка; ✅Использованы хедеры и футеры там, где это необходимо согласно дизайну ✅Студент демонстрирует понимание принципов делегата и Data source.	Реализовано кастомное поведение для коллекций, такое как pull-to-refresh, swipe-to- delete и т.д. – 🍏	Хорошо, +использован DiffableDataSource
4. Верстка коллекции	✅ Ячейки свёрстаны корректно, содержимое не вылезает за границы ячейки; ✅Корректно реализовано обновление и нет падений при обновлении таблицы; ✅Правильно отображается содержание пустой коллекции; 🙌Пагинация в необходимых местах работает корректно — нет множественных запросов с одними и теми же параметрами, показывается «крутилка».	Кастомный layout – 🍏Реализовано кастомное поведение для коллекций, такое как pull-to-refresh, swipe-to- delete и т.д. – 🍏	Верстка ok, пустые экраны тоже ок, обновляется все корректно. Пагинация избыточна, ~~но мне не хватило крутилки при загрузке данных по коллекции.~~
5. Соответствие макету Figma	🙌Экраны и навигация выполнены верно. Допущение: В одном-двух местах во всём проекте шрифт может быть неправильного размера, картинка криво экспортирована	Правильный размер и сам шрифт, правильные картинки и разрешение в них, правильная навигация между экранами – 🍏 «Пиксель пёрфект» попадание по дизайну – макеты и вёрстка идеально совпадают – 🍏	Верстка практически идеальная у всех элементов – тут я ставлю «Пиксель пёрфект» заочно. ~~Но: навигационный бар у экранов не соответствует макетам. Нужно исправить.~~
6. UIKit, Foundation	✅UI обновляется только в главном потоке; ✅При сетевых запросах блокируется интерфейс; ✅Почти не используются depricated методы (не более одного-двух раз за весь проект); ✅Для работы с датами использован date formatter. ✅Нет retain cycles и утечек памяти.	Когда пользователь переходит на следующий экран при множественном нажатии на кнопку или элемент интерфейса не происходит множественного открытия экрана, на который переходят. – 🍏	Отлично! Комментарий: утечек не вижу, но обратила внимание как быстро растет используемая память при открытии экранов. Ты загружаешь много картинок, возможно нужно уменьшить размер кэша у Kingfisher .
7. Работо-способность приложения	✅ Проект компилируется, допустимо минимальное количество некритичных предупреждений (не приводят к падению или некорректной работе приложения). Максимум 3 предупреждения. 🙌Нет падений при разных входных данных; ❌Приложение не содержит критических багов, которые могут привести к падению или неработоспособности; ❌Баги, которые возникают, не влияют на работу приложения в целом и не приводят к нарушению его логики.	Приложение не содержит никаких багов. – 🍏	~~В этом пункте большой недобор: не реализована часть с сохранением избранных коллекций и помещения в корзину (PUT-методы)~~ ~~При открытии подряд нескольких элементов приложение падает из-за assertFailure – не обработанная ошибка.~~ При возникновении ошибок при загрузке коллекции(множественные заходы в разные колекции, пока мок сервер не перестанет отвечать), можно попасть на пустой экран (ни коллекции, ни сообщения что коллекция пуста), и уйти с него нельзя.
8. Архитектура. Одна из MVP, MVC, MVVM	✅В работе компоненты архитектуры имеют правильные «ответсвенности». Например, не названо ViewModel то, что должно быть презентером. Небольшие кусочки кода могут остаться во вью и при этом не переехать в презентер ✅Последовательно использует выбранную архитектуру. Экраны содержат все заявленные архитектурой элементы. ✅Правильно называется классы, продемонстрированы на практике связи между компонентами архитектуры.	В работе используются конечные автоматы. – 🍏	В целом ViewModel и View занимаются своими вещами – нет критичных ошибок, хотя есть еще несколько не самых лучших мест.
9. Работа с сетевыми запросами	🙌Реализована обработка ошибок, таких как отсутствие интернет-соединения, невалидные данные и другие. 🙌Реализованы различные типы запросов, такие как GET, POST, у некоторых эпиков PUT, DELETE		~~Нет важного запроса на изменение коллекции – PUT, не все ошибки обработаны корректно (со своим экраном или как минимум алертом).~~ Теперь все ок

Компоненты	Базовый вариант	Великолепный вариант	Комментарий
10. Использование многопоточности	✅ В работе определены потенциальные race conditions и реализована защита от них. ✅ Реализована обработка ответов на сетевые запросы в фоновом потоке ✅ Реализовано правильное использование многопоточности, чтобы избежать race conditions и deadlocks.	Реализовано кэширование ответов на запросы, чтобы уменьшить количество сетевых запросов. – 🍏	Отлично!
11. Хранение данных	✅ Реализована работа с локальным хранилищем, таким как UserDefaults или Keychain.		Продемонстрировано на сохранении вида сортировки.
12. Автотетестирование	Автотестов нет	Используются разнообразные типы тестов 🍏 Все критические участки полностью (бизнес-логика) покрыты тестами; 🍏 покрытие достаточно полное, тесты запускаются и работают быстро; 🍏 Есть скриншотные и Unit-тесты. 🍏	
13. Чистота кода	✅ В большинстве случаев корректно названы переменные (понятно и логично), ✅ используется единый принцип именования camelCase, ✅ код сервисов вынесен в отдельные классы, ✅ нет серьёзных нарушений принципов SOLID, KISS, DRY, YAGNI; ✅ Используются паттерны, ✅ функции нормального размера с нормальными названиями; ✅ Дублирование кода сведено к минимуму; ✅ Корректно расставлены отступы, нет лишних неинформативных комментариев; ✅ Корректная работа с классами и структурами (нет ошибок с референс и Велью семантикой) ✅ нет неиспользуемого кода; ✅ Не используется Forced unwrap.	Корректно названы переменные, код сервисов вынесен в отдельные классы, 🍏 Корректная работа с классами и структурами), 🍏 Использует SwiftLint 🍏	По базовому варианту отлично! Рекомендация настроить SwiftLint - достаточно много ворнингов и даже одна критичная ошибка генерируется при запуске проекта.
14. Работа с Git	✅ Названия коммитов понятны и логичны ✅ Помечена связь с тасками в таск- трекере в описании пулл-реквеста; ✅ Прописана выбранная архитектура и способ вёрстки.	Адекватный размер коммитов 🍏	Общий комментарий по работе с гитом: старайся выносить в название комита больше конкретики. Именно название комита видно в общем списке. Здесь стараются писать от глагола и что изменилось. Сообщение с комментарием тоже хорошо – но его не видно в большинстве менеджеров. Кроме того, рекомендую вернуться к теме Git – название ветки получилось странным (ожидается что-то вроде feature/catalogue)
Подведение итогов	Есть еще баг при загрузке коллекций в режиме стресс тестирования: много быстрых открытий друг за другом экранов.	7 🍏 из 18	Работа на хорошем уровне. Мне кажется не хватило софтскилов уточнить требования про реализацию задания: очень обидно, что придется доделывать пару крупных фич в режиме аврала (очень похоже на жизнь) Рекомендация: всегда читать ТЗ и задавать вопрсоы заказчику, сомневаться в полноте требований и своей реализации. Продуктовые задачи решены не полностью, хотя выглядит в качестве прототипа очень хорошо. ~~Необходимо: доправить оставшиеся проблемы. Вернуть работу на еще одно ревью, с исправленными проблемами.~~ Рекомендация придерживаться модели работы с ветками gitflow или его упрощенные версии. Работа принята для мерджа с эпиками команды.