Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Отчет по курсовой работе

По курсу «Конструирование программного обеспечения»

Выполнили Ершов С.С. ,Спиваков М.Б.,
студенты гр. 5130904/20105 и 5130904/20103 Косминская О.А., Горохов А.В.

Преподаватель Юркин В. А.

Санкт-Петербург
2025 г.

Содержание

Определение проблемы

В компании друзей при организации совместных поездок или других мероприятий, требующих общих денежных затрат, бывает сложно отследить совместные траты и долги друг другу при оплате одной картой за всех. Пользователи вынуждены тратить много времени на подсчет совместных трат и выявление задолженностей, что неудобно и увеличивает вероятность ошибиться и чего-то не учесть. Задача кратно усложняется при большом количестве пользователей, а также при поездках в другие страны и использовании различных валют.

Выработка требований

	Пользовательская история
1	Я, как пользователь, хочу создавать группы и добавлять участников, чтобы разделять расходы между ними
2	Я, как пользователь, хочу сканировать QR-код на чеке, чтобы автоматически добавлять информацию о покупке в историю расходов
3	Я, как пользователь, хочу добавлять расходы в разных валютах и автоматически конвертировать их в выбранную валюту группы. Конвертация происходит в момент добавления.
4	Я, как пользователь, хочу экспортировать данные о расходах в формате CSV или pdf для дальнейшего анализа, создание документа по запросу

Примерная оценка числа пользователей

Число активных пользователей в месяц (MAU) 50000000 штук

Число активных пользователей в день (DAU) 3000000 штук

Примерная оценка периода хранения информации

От года до 5 лет:

№	Характеристика	Подраздел	Значение	Единица измерения
12	Объём хранения данных в РСУБД для таблицы users	год	205	Гб
		три года	615	Гб
		5 лет	1025	Гб
13	Объём хранения данных в РСУБД для таблицы groups	год	1957	Мб
		три года	5871	Мб
		5 лет	9785	Мб
14	Объём хранения данных в РСУБД для таблицы expense	год	38	Гб
		три года	114	Гб
		5 лет	190	Гб
15	Объём хранения данных в РСУБД для таблицы userExpense	год	1177	Мб
		три года	3531	Мб
		5 лет	5885	Мб

Разработка архитектуры и детальное проектирование

Характер нагрузки на сервис:

- Соотношение R/W нагрузки и объемы трафика

№	Характеристика	Подраздел	Значение	Единица измерения
1	Число активных пользователей в месяц (MAU)		50000000	штук
2	Число активных пользователей в день (DAU)		3000000	штук
3	(Ключевая операция) Создание группы	В день	1285714	штук
		В час	53571,41667	штук
		В секунду	14,88094907	штук
4	(Ключевая операция) Добавление нового расхода (как и вручную, так и сканированием QR)	В день	2571428	штук
		В час	107142,8333	штук
		В секунду	29,76189815	штук
5	(Ключевая операция) Экспорт данных о расходах группы в формате CSV и PDF	В день	1500000	штук
		В час	1,24007909	штук
		В секунду	17,36111111	штук
6	Активные часы пользователя		24	час
7	Write-трафик для Ключевой операции "Создание группы" всеми пользователями	В секунду	0,02976189815	Мбит
8	Write-трафик для Ключевой операции "Добавление нового расхода" всеми пользователями	В секунду	0,03571427778	Мбит
9	Read-трафик для ключевой операции " Экспорт данных о расходах группы в формате CSV и PDF"	В секунду	35,55555556	Мбит
10	Число CDN Серверов для оптимизации Read-трафика		6	штук
11	Соотношение read/write в системе		35	35 к 1

Предположительные значения и дальнейшие расчёты нагрузки основаны на соображениях о том, что группы создаются реже, чем просто используются (добавление новых расходов или просмотр данных)

- Объемы дисковой системы

№	Характеристика	Подраздел	Значение	Единица измерения
12	Объём хранения данных в РСУБД для таблицы users	год	205	Гб
		три года	615	Гб

		5 лет	1025	Гб
13	Объём хранения данных в РСУБД для таблицы groups	год	1957	Мб
		три года	5871	Мб
		5 лет	9785	Мб
14	Объём хранения данных в РСУБД для таблицы expense	год	38	Гб
		три года	114	Гб
		5 лет	190	Гб
15	Объём хранения данных в РСУБД для таблицы userExpense	год	1177	Мб
		три года	3531	Мб
		5 лет	5885	Мб

Первые две диаграммы из подхода https://c4model.com/

Рисунок 1 Контекстная диаграмма C4 для приложения SplitWallet

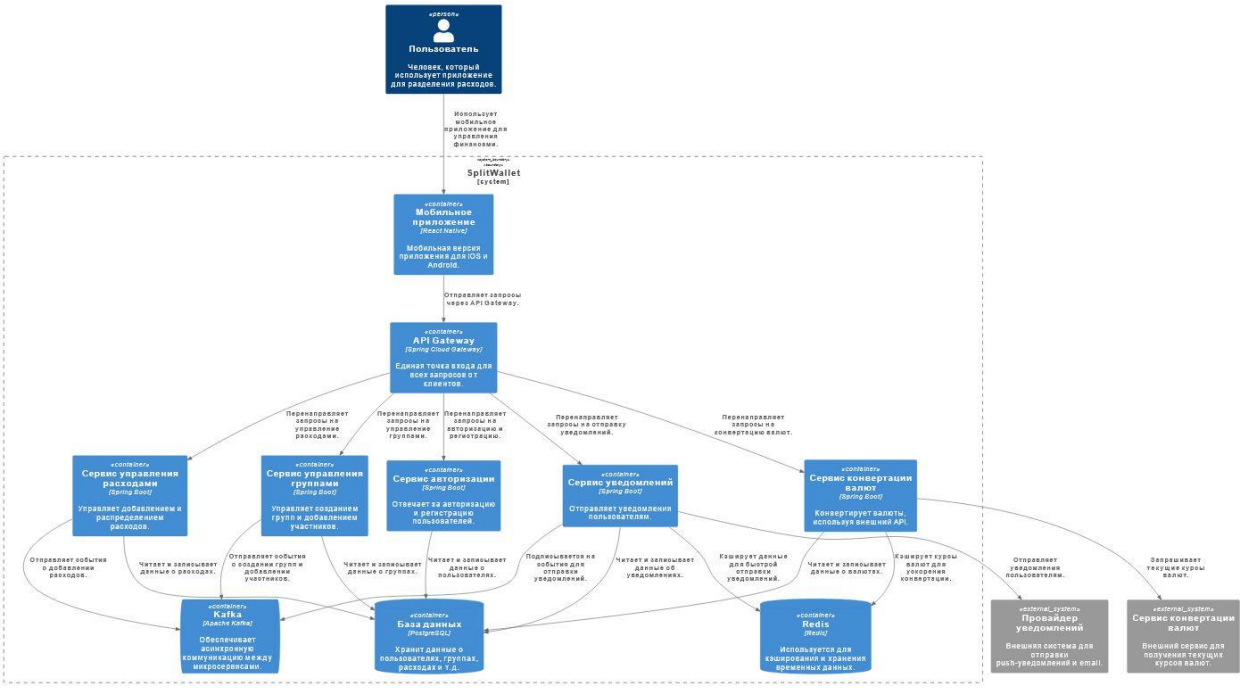


Рисунок 2 Диаграмма контейнеров C4 для приложения SplitWallet

Контракты API

(реализация запросов при помощи swagger)

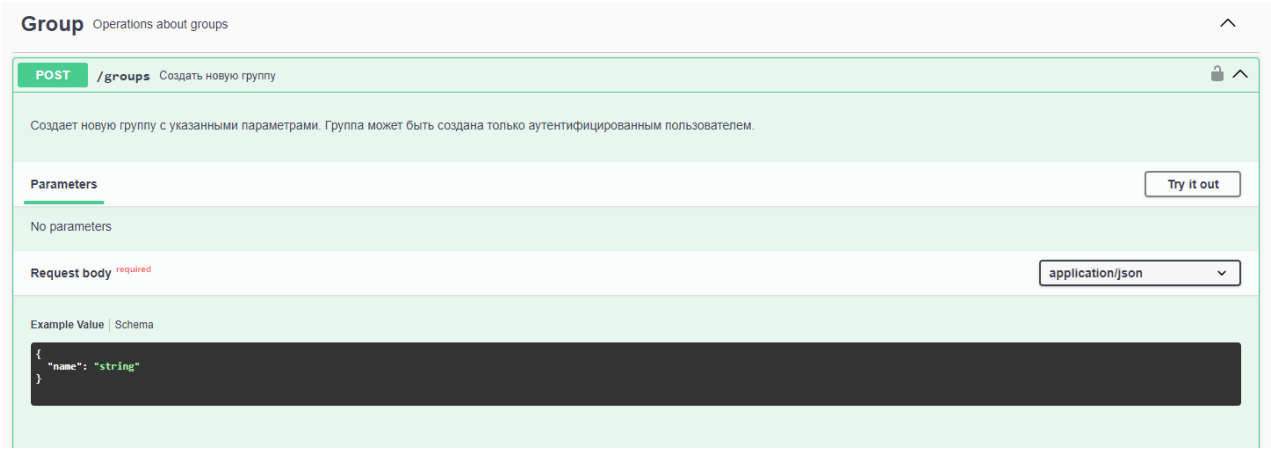


Рисунок 3 Пример запроса по созданию новой группы

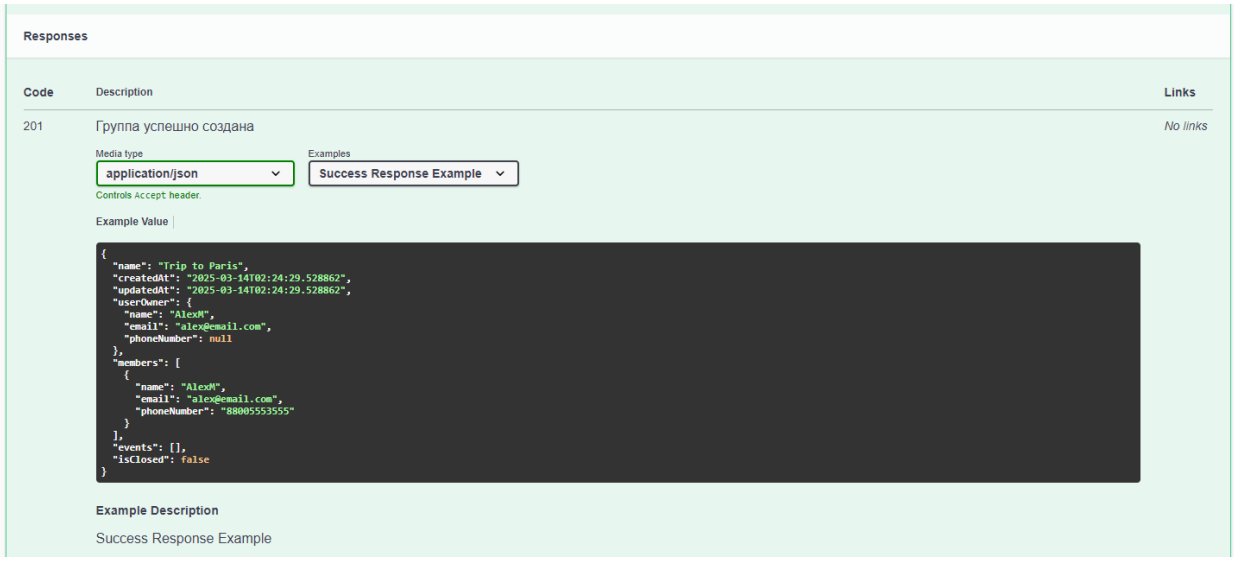


Рисунок 4 Пример ответа с кодом 201 на запрос по созданию новой группы

Рисунок 5 Пример ответа с кодом 400 на запрос по созданию новой группы

Рисунок 6 Пример ответа с кодом 500 на запрос по созданию новой группы

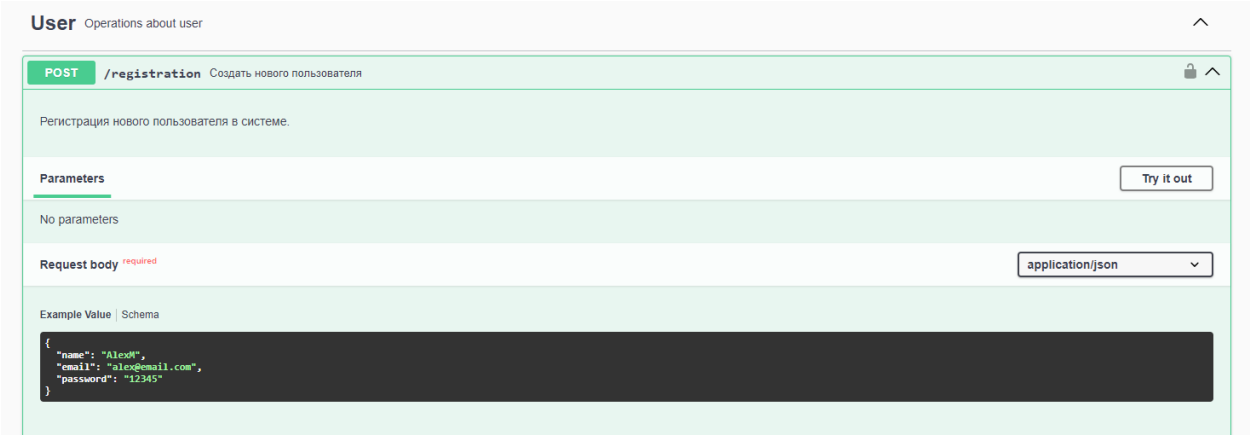


Рисунок 7 Пример запроса для регистрации нового пользователя в системе

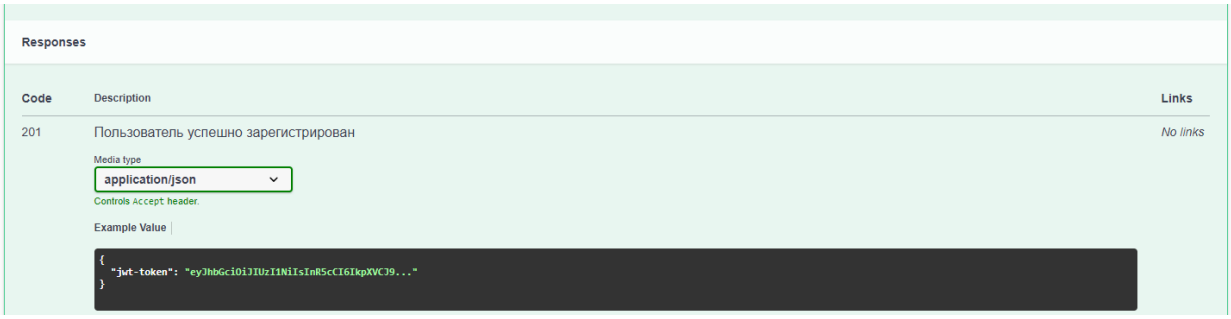


Рисунок 8 Пример ответа с кодом 201 на запрос для регистрации нового пользователя в системе

Рисунок 9 Пример ответа с кодом 400 на запрос для регистрации нового пользователя в системе

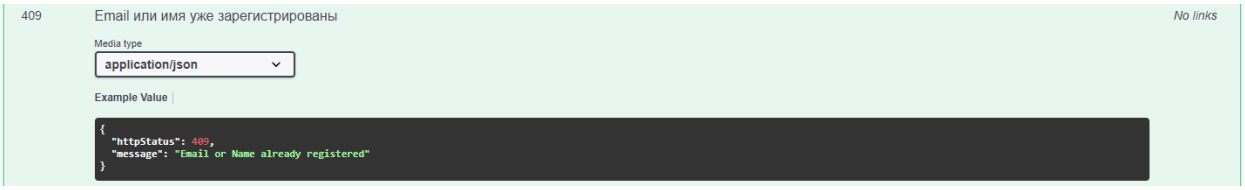


Рисунок 10 Пример ответа с кодом 409 на запрос для регистрации нового пользователя в системе

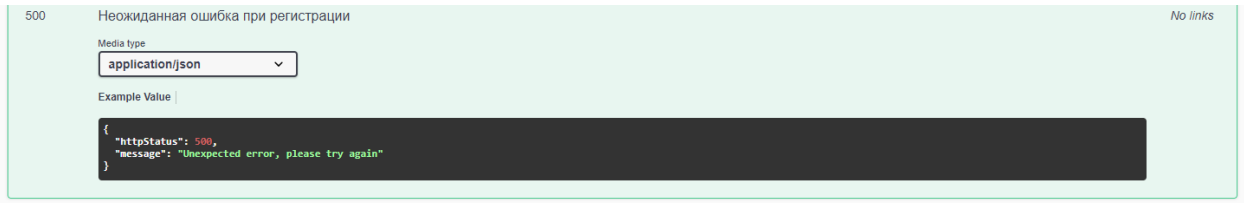


Рисунок 11 Пример ответа с кодом 500 на запрос для регистрации нового пользователя в системе

Рисунок 12 Пример запроса для аутентификации пользователя с использованием логина и пароля

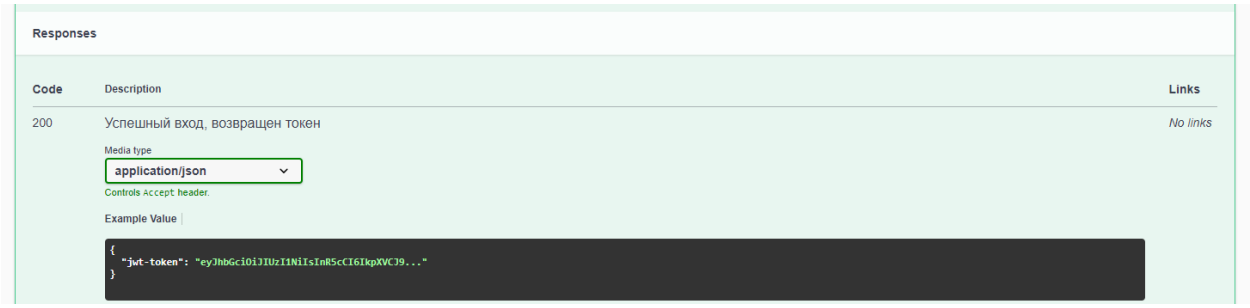


Рисунок 13 Пример ответа с кодом 200 на запрос для аутентификации пользователя с использованием логина и пароля

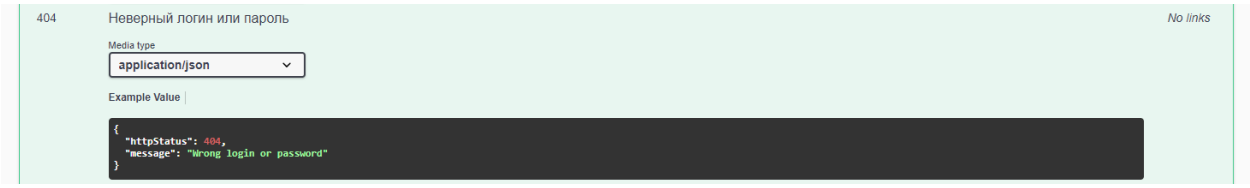


Рисунок 14 Пример ответа с кодом 404 на запрос для аутентификации пользователя с использованием логина и пароля

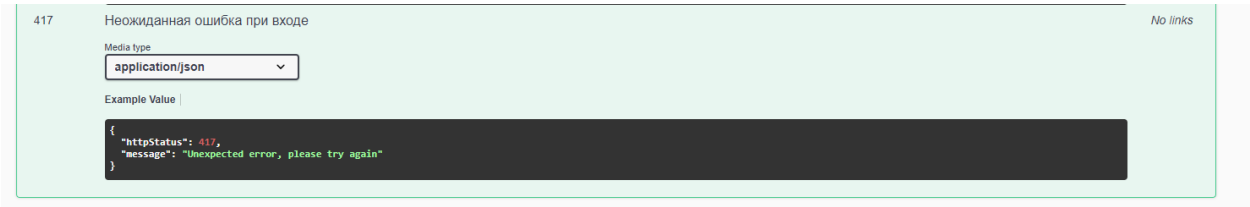


Рисунок 15 Пример ответа с кодом 417 на запрос для аутентификации пользователя с использованием логина и пароля

Ожидаемые нефункциональные требования на время отклика

	Бизнес-требование	Нефункциональное требование
1	Приложение должно поддерживать сканирование QR-кода на чеке для автоматического добавления расходов	Формат поддерживаемых QR-кодов: стандартные QR-коды для чеков. Время обработки QR-кода: не более 2 секунд
2	Приложение должно позволять создавать сводку в разных форматах при закрытии группы	Время экспорта: не более 5 секунд.
3	Приложение должно поддерживать уведомления о новых расходах и изменениях в группах	Время доставки уведомлений: от 5 секунд до минуты. Поддержка push-уведомлений и email-уведомлений. Уведомление приходит, только если это затрагивает пользователя
4	Приложение должно поддерживать оффлайн-режим для добавления расходов	Максимальное время синхронизации: не более 10 секунд после восстановления соединения. Создание группы только при наличии интернет соединения
5	Приложение должно предоставлять возможность быстрой авторизации по номеру телефона для незарегистрированных пользователей	Время авторизации: не более 5 секунд. Поддержка SMS-кода для подтверждения номера. Максимальное время доставки SMS: 10 секунд
6	Приложение должно предоставлять возможность быстрой авторизации для незарегистрированных пользователей	Время авторизации: не более 5 секунд. Поддержка кода для подтверждения почты. Максимальное время доставки сообщения: 10 секунд. Использовать Keycloak для реализации SSO и поддержки авторизации через Google и VK.

Схема базы данных

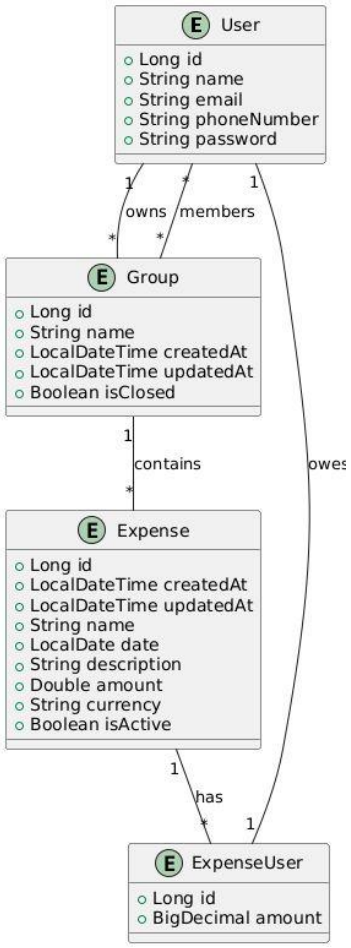


Рисунок 16 Схема базы данных (концептуальная модель данных)

Схема масштабирования сервиса при росте нагрузки в 10 раз

При росте нагрузки можно будет увеличить количество серверов для хранения нужных данных. Также потребуется увеличить объём кэша для хранения временных данных. Возможно, понадобится оптимизировать обмен запросами, чтобы уменьшить время и увеличить пропускную способность.

Кодирование и отладка

В процессе данного этапа были реализованы серверная и Android- части приложения. Примеры работы приложения.

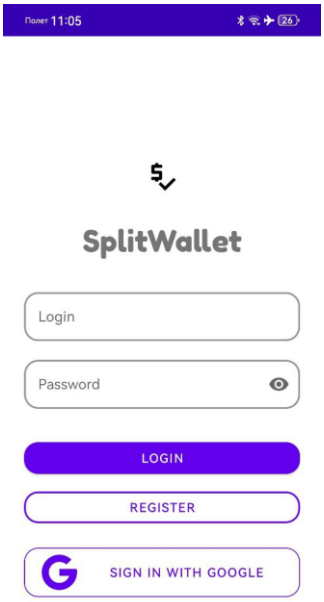


Рисунок 17 Страница входа в приложение

При успешном входе/регистрации пользователь попадает на домашнюю страницу.

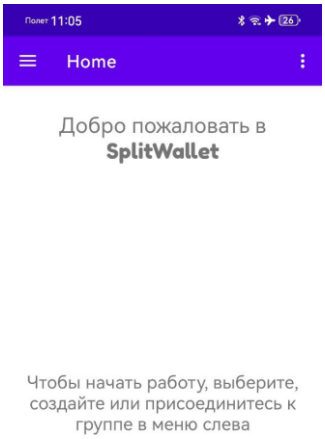


Рисунок 18 Домашняя страница приложения

При нажатии на знак в левом верхнем углу пользователь может открыть боковое меню, содержащее логин и email пользователя, а также кнопки для перехода на первую страницу, создания группы, присоединения к группе и список групп с кнопками для их удаления:

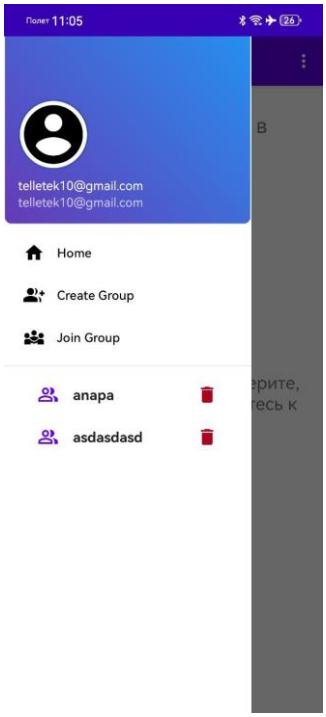


Рисунок 19 Боковое меню

При нажатии на кнопку для создания группы открывается диалоговое окно для ввода названия.

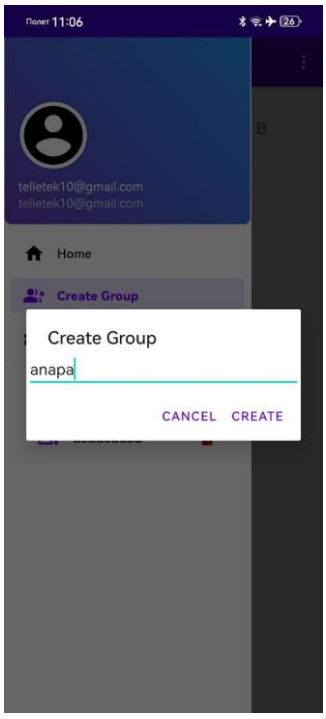


Рисунок 20 Окно создания группы

После нажатия кнопки “Create” в этом окне (при непустом названии группы) новая группа создаётся и появляется у пользователя в боковом меню.

При нажатии в боковом меню кнопки для присоединения к группе (“Join Group”) открывается страница присоединения к группе: пользователь должен ввести шестизначный код, полученный от одного из участников данной группы.

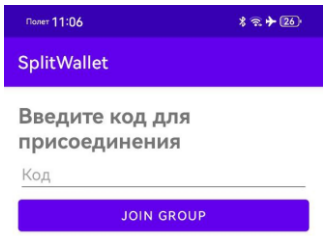


Рисунок 21 Страница присоединения к группе

После присоединения к группе она появляется у пользователя в боковом меню.

Теперь рассмотрим пример страницы группы. На ней есть вкладки с расходами (“Expences”) и с более детальной информацией об участниках (“Details”).

На вкладке с расходами есть каждый расход: его название, имя добавившего участника, сумма, валюта, дата создания и описание.

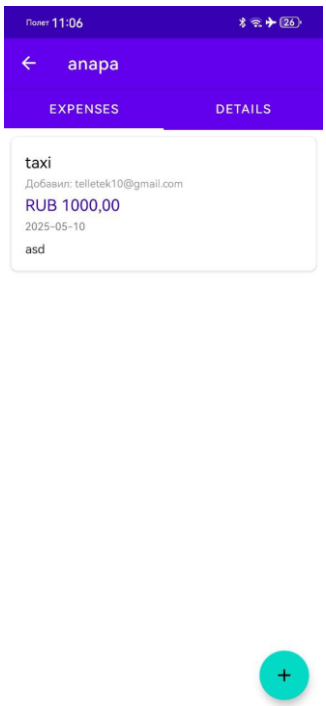


Рисунок 22 Вкладка страницы группы с расходами

Также тут есть кнопка («+») для добавления расхода вручную или с помощью сканера QR-кода.

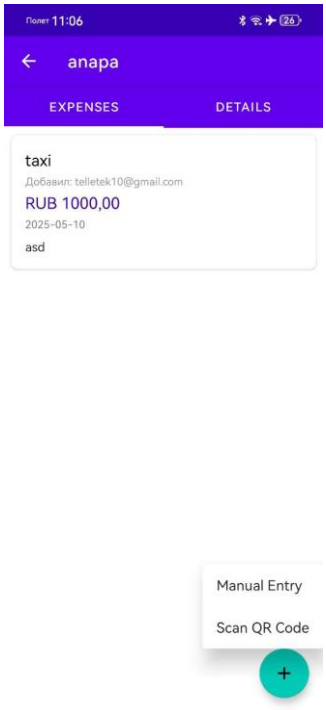


Рисунок 23 Вкладка страницы группы с расходами и кнопкой для добавления нового расхода

При нажатии на кнопку для добавления расхода вручную пользователь видит следующее окно, содержащее поля для ввода названия расхода, его описания, суммы, даты (автоматически выставляется на текущий день, но можно изменить) и валюты.

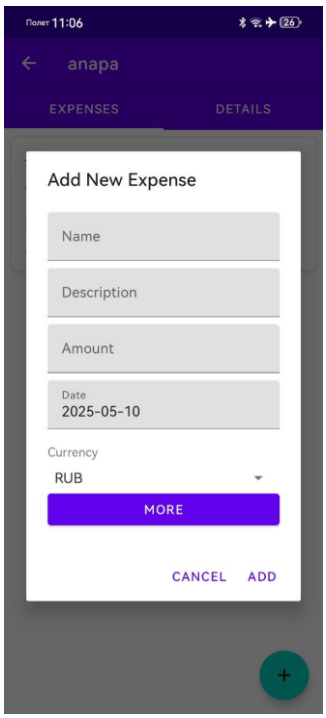


Рисунок 24 Окно для добавления нового расхода

Также в этом окне есть кнопка “More”. Она открывает окно, позволяющее изменить распределение суммы расхода между его участниками. По умолчанию вся сумма присваивается создателю расхода (соответственно, у остальных она тогда равна 0). Можно ввести новое распределение или исключить каких-то участников из расчёта распределения данного расхода. Однако сумма расхода должна совпадать с изначальной.

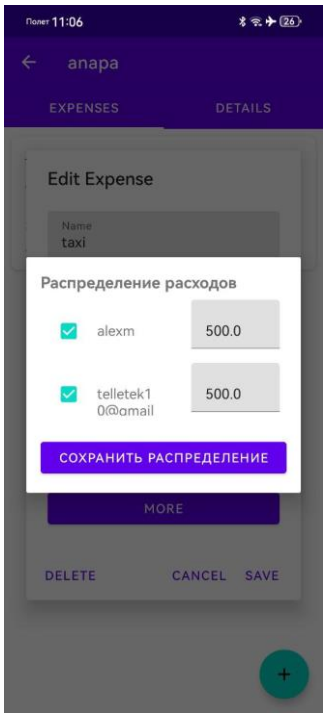


Рисунок 25 Окно для настройки распределения расходов

При появлении нового расхода в группе, изменении соотношения расходов в группе, а также при изменениях в группах пользователям приходят push-уведомления.

При нажатии второй кнопки и сканировании QR-кода поля информации с суммой и датой заполняются автоматически, а в качестве названия пишется «Чек № …».

На вкладке с информацией об участниках группы есть список участников с логином, email и балансом (количеством денег, которые по расходам ему должны другие участники).

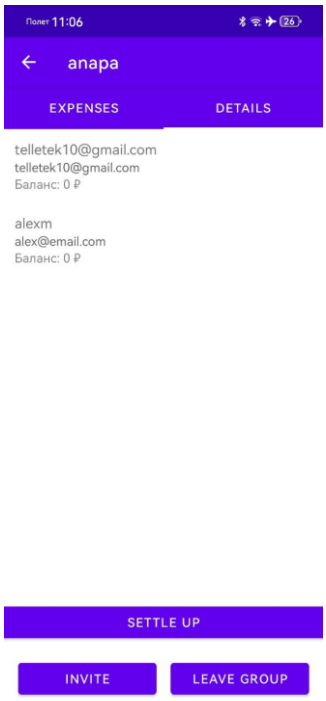


Рисунок 26 Вкладка страницы группы с информацией об участниках

Также есть кнопки для приглашения в группу нового участника (“Invite”) и выхода из группы (“Leave Group”).

При нажатии кнопки для приглашения в группу открывается окно с кодом для присоединения, который другой пользователь должен ввести в поле на странице “Join Group”.

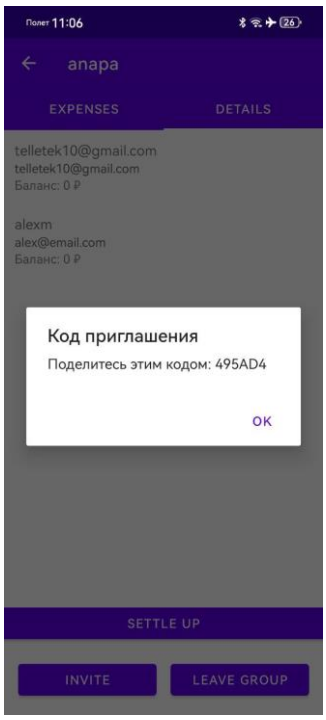


Рисунок 27 Окно с кодом для присоединения к группе

При нажатии кнопки для выхода из группы (“Leave Group”) открывается окно с подтверждением выхода:

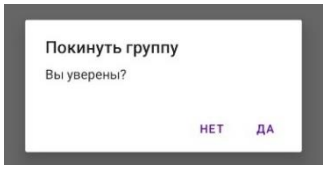


Рисунок 28 Окно с подтверждение выхода из группы

После выхода из группы она исчезает из бокового меню у пользователя. Чтобы вернуться в группу, ему следует снова ввести код для вступления.

Кнопка “Settle Up” открывает окно с информацией о долгах, связанных с текущим пользователем в данной группе – сколько он должен и сколько ему должны - и кнопкой для создания отчёта (“Create Report”) в формате PDF.

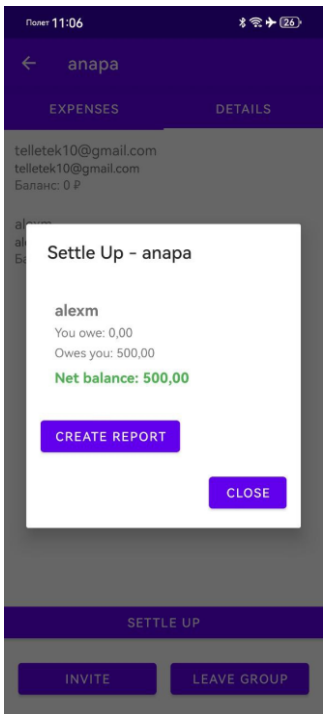


Рисунок 29 Окно "Settle Up" для группы с текущей информацией и кнопкой для создания отчёта

При нажатии кнопки “Create Report” начинается процесс генерации отчёта:

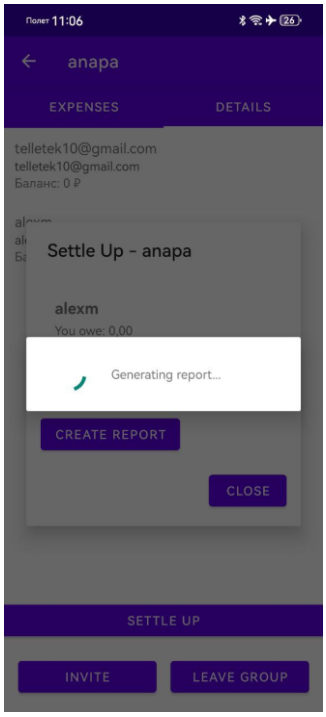
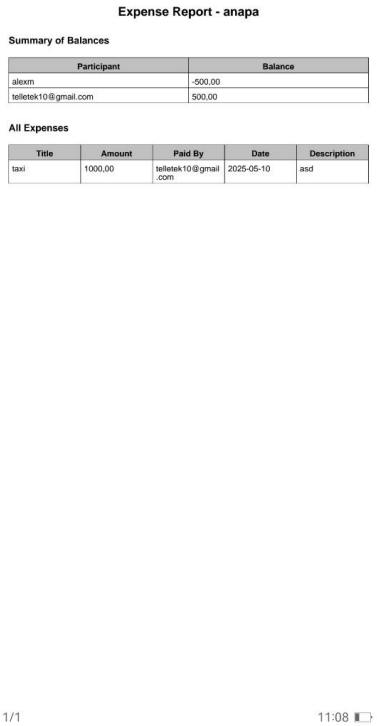


Рисунок 30 Процесс генерации отчёта

По окончании генерации пользователю предоставляется отчёт в следующем виде:



Expense Report - anapa

Summary of Balances

Participant	Balance
alexm	-500,00
telletek10@gmail.com	500,00

All Expenses

Title	Amount	Paid By	Date	Description
taxi	1000,00	telletek10@gmail.com	2025-05-10	asd

1/1 11:08

Рисунок 31 Отчёт о расходах пользователя по группе в формате PDF

В заголовке написаны названия отчёта и группы. В первой таблице (“Summary of Balances”) указаны участники расхода (“Participant”) и суммы денег, которые они должны получить или заплатить. Вторая таблица (“All Expenses”) содержит полную информацию о расходах: название, сумма, логин пользователя, который платит, дата и описание.

Создатель группы имеет право её удалить, а приглашённый участник – только покинуть. При нажатии кнопки для удаления группы напротив её названия в боковом меню откроется следующее окно:

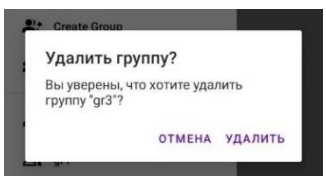


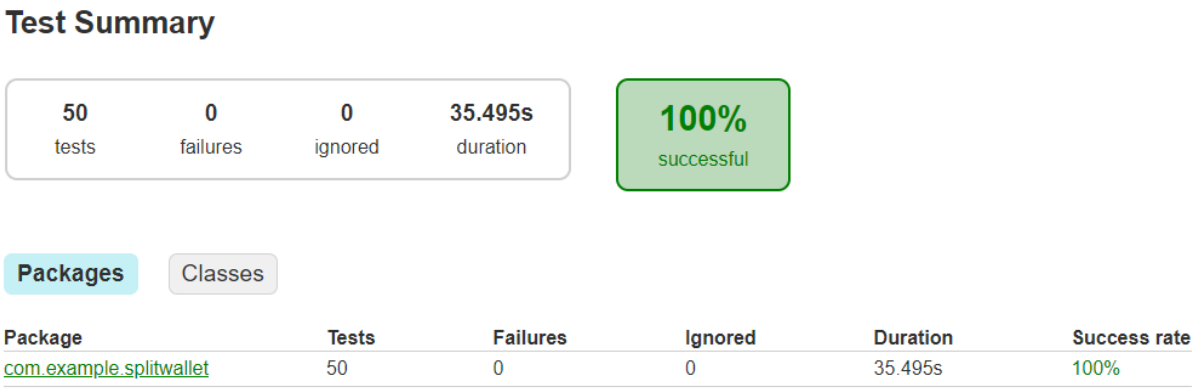
Рисунок 32 Окно подтверждения удаления группы

После удаления она исчезает из бокового меню у всех участников.

Unit-тестирование

Были написаны unit-тесты для классов и методов, где применяются простые зависимости или требуется протестировать исключения и граничные случаи.

Была получена статистика по выполнению тестов. Из 50 написанных юнит-тестов все были выполнены успешно примерно за 35,5 секунд.



Test Summary

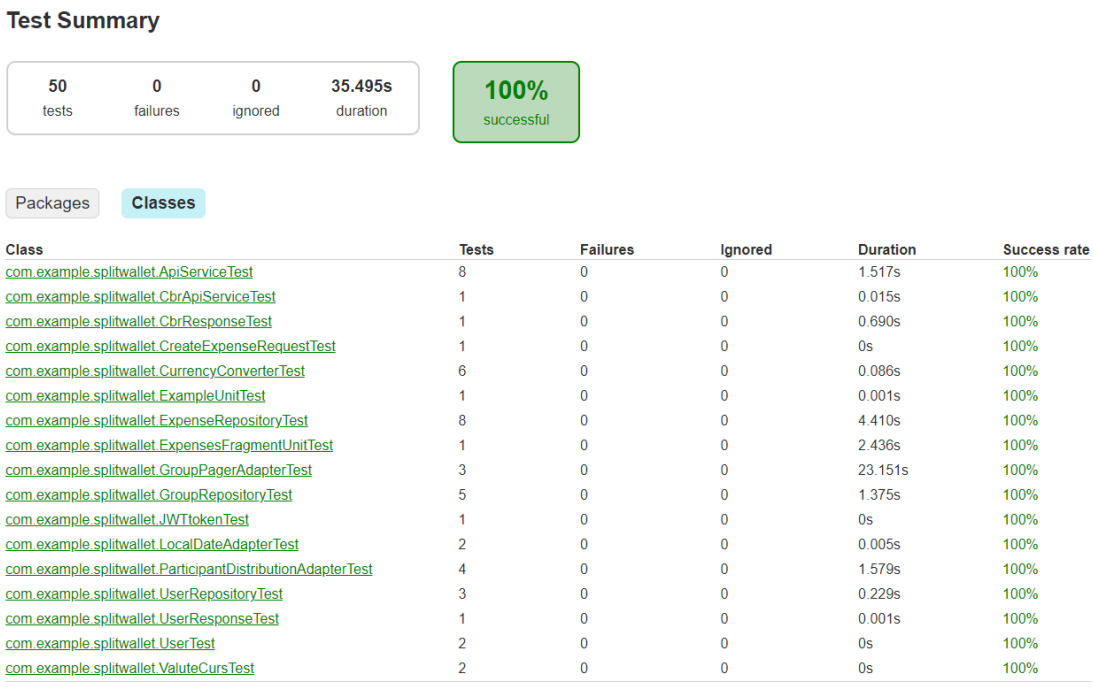
50 tests	0 failures	0 ignored	35.495s duration	100% successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.example.splitwallet	50	0	0	35.495s	100%

Рисунок 33 Полученная статистика по выполнению юнит-тестов

Полученная статистика тестов по классам:



Test Summary

50 tests	0 failures	0 ignored	35.495s duration	100% successful

Packages | Classes

Class	Tests	Failures	Ignored	Duration	Success rate
com.example.splitwallet.ApiServiceTest	8	0	0	1.517s	100%
com.example.splitwallet.CbrApiServiceTest	1	0	0	0.015s	100%
com.example.splitwallet.CbrResponseTest	1	0	0	0.690s	100%
com.example.splitwallet.CreateExpenseRequestTest	1	0	0	0s	100%
com.example.splitwallet.CurrencyConverterTest	6	0	0	0.086s	100%
com.example.splitwallet.ExampleUnitTest	1	0	0	0.001s	100%
com.example.splitwallet.ExpenseRepositoryTest	8	0	0	4.410s	100%
com.example.splitwallet.ExpensesFragmentUnitTest	1	0	0	2.436s	100%
com.example.splitwallet.GroupPagerAdapterTest	3	0	0	23.151s	100%
com.example.splitwallet.GroupRepositoryTest	5	0	0	1.375s	100%
com.example.splitwallet.JWTtokenTest	1	0	0	0s	100%
com.example.splitwallet.LocalDateAdapterTest	2	0	0	0.005s	100%
com.example.splitwallet.ParticipantDistributionAdapterTest	4	0	0	1.579s	100%
com.example.splitwallet.UserRepositoryTest	3	0	0	0.229s	100%
com.example.splitwallet.UserResponseTest	1	0	0	0.001s	100%
com.example.splitwallet.UserTest	2	0	0	0s	100%
com.example.splitwallet.ValuteCursTest	2	0	0	0s	100%

Рисунок 34 Полученная статистика по выполнению тестов по классам

Среднее время выполнения тестов: 0,7099.

Рассмотрим наглядное соотношение количества тестов для разных классов. Можем наблюдать наибольшее количество тестов - для классов типа Repository, а также ApiService, где тестируются нужные запросы. Наименьшее количество тестов в данном случае, по большей части, у классов, где есть только конструктор или какая-либо инициализация.

Рисунок 35 Соотношение количества юнит-тестов по классам

Рассмотрим также соотношение времени выполнения тестов относительно классов.

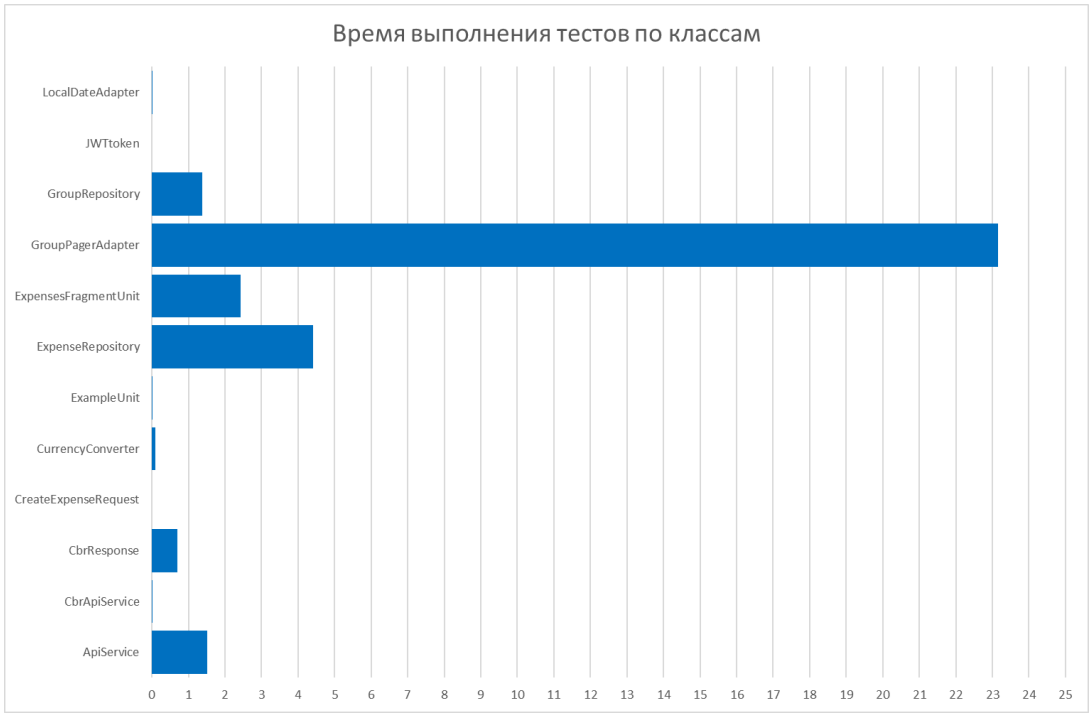


Рисунок 36 Соотношение времени выполнения юнит-тестов по классам

Однако на этой диаграмме плохо видны (или не видны) меньшие значения из-за большого разброса показателей. Поэтому иллюстрируем динамику изменения времени выполнения в зависимости от класса другим способом.

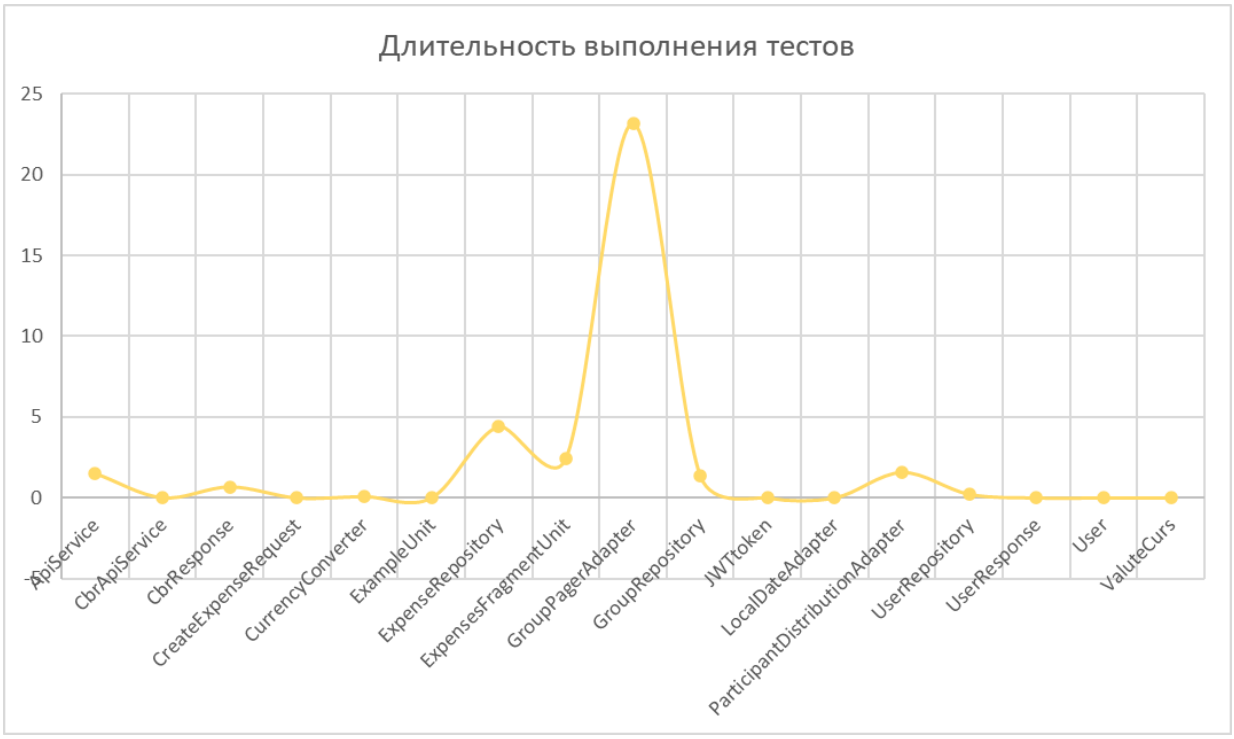


Рисунок 37 График зависимости времени выполнения юнит-тестов от класса

Можем наблюдать, что больше всего времени занимают тесты для GroupPagerAdapter, хотя их количество не является максимальным из рассматриваемых

Кроме того, был проведён анализ покрытия тестами определённых классов и методов, для которых они требовались.

Element	Class, %	Method, %	Line, %	Branch, %
models	51% (17/33)	24% (31/126)	18% (126/681)	26% (51/190)
Balance	0% (0/1)	0% (0/1)	0% (0/6)	100% (0/0)
BalanceAdapter	0% (0/2)	0% (0/6)	0% (0/23)	0% (0/4)
CbrResponse	100% (6/6)	100% (6/6)	100% (6/6)	100% (0/0)
CreateExpenseRequest	100% (1/1)	100% (5/5)	100% (5/5)	100% (0/0)
CreateGroupRequest	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
CurrencyConverter	50% (1/2)	62% (5/8)	63% (53/83)	69% (32/46)
Expense	0% (0/1)	100% (0/0)	100% (0/0)	100% (0/0)
ExpenseAdapter	0% (0/3)	0% (0/13)	0% (0/39)	0% (0/10)
ExpenseCallback	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
ExpensesCallback	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
ExpensesFragment	100% (1/1)	2% (1/40)	2% (7/307)	4% (3/64)
ExpenseUser	100% (1/1)	100% (1/1)	100% (5/5)	100% (0/0)
GoogleLoginRequest	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)
Group	0% (0/1)	100% (0/0)	100% (0/0)	100% (0/0)
GroupBalancesResponse	0% (0/1)	100% (0/0)	100% (0/0)	100% (0/0)
GroupDetailsFragment	0% (0/2)	0% (0/20)	0% (0/117)	0% (0/44)
GroupPagerAdapter	0% (0/1)	0% (0/3)	0% (0/8)	0% (0/2)
JWTtoken	100% (1/1)	100% (1/1)	100% (2/2)	100% (0/0)
LocalDateAdapter	100% (1/1)	100% (2/2)	100% (2/2)	100% (0/0)
LoginRequest	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
ParticipantDistributionAdapter	50% (2/4)	38% (5/13)	52% (31/59)	80% (16/20)
RegisterRequest	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
UpdateExpenseRequest	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
User	100% (1/1)	100% (3/3)	100% (8/8)	100% (0/0)
UserResponse	100% (1/1)	100% (1/1)	100% (6/6)	100% (0/0)
ValuteCurs	100% (1/1)	100% (1/1)	100% (1/1)	100% (0/0)
repository	31% (5/16)	19% (12/61)	18% (31/164)	6% (2/32)
ExpenseRepository	83% (5/6)	48% (12/25)	37% (31/82)	20% (2/10)
GroupRepository	0% (0/7)	0% (0/26)	0% (0/55)	0% (0/14)
UserRepository	0% (0/3)	0% (0/10)	0% (0/27)	0% (0/8)

Рисунок 38 Анализ покрытия unit-тестами

В данном случае практически не учитывается покрытие для классов GroupPagerAdapter, GroupRepository и UserRepository (GroupPagerAdapterTest, GroupRepositoryTest и UserRepositoryTest), так как при выполнении их в обычном режиме тесты проходят, а при запуске с анализом покрытия проявляется несоответствие версий Robolectric с текущей версией Android SDK. Однако смена версий не даёт результатов, а в обычном режиме тесты выполняются корректно, поэтому перечисленные классы тоже можно считать протестированными.

Test Summary

50	11	0	24.222s
tests	failures	ignored	duration

78% successful

Failed tests Packages Classes

GroupPagerAdapterTest. testCreateFragment_invalidPosition_throwsException
GroupPagerAdapterTest. testCreateFragment_returnsCorrectFragment
GroupPagerAdapterTest. testItemCount_returnsTwo
GroupRepositoryTest. createGroup_failure
GroupRepositoryTest. createGroup_success
GroupRepositoryTest. deleteGroup_failure
GroupRepositoryTest. deleteGroup_success
GroupRepositoryTest. getUserGroups_success
UserRepositoryTest. login_failed
UserRepositoryTest. login_networkError
UserRepositoryTest. login_successful

Рисунок 39 Тесты, которые не прошли при другом типе запуска

Интеграционное тестирование

Было проведено интеграционное тестирование, в рамках которого были реализованы тесты на взаимодействие различных частей системы по определённым сценариям, реализованы некоторые пользовательские сценарии.

Была получена статистика по выполнению тестов. Из 39 написанных интеграционных тестов все были выполнены успешно примерно за 18 секунд.

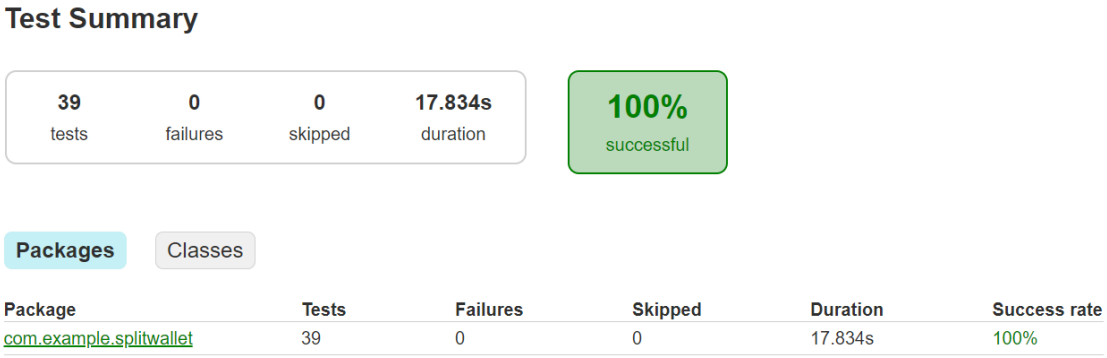


Рисунок 40 Полученная статистика по выполнению интеграционных тестов

Полученная статистика тестов по классам:

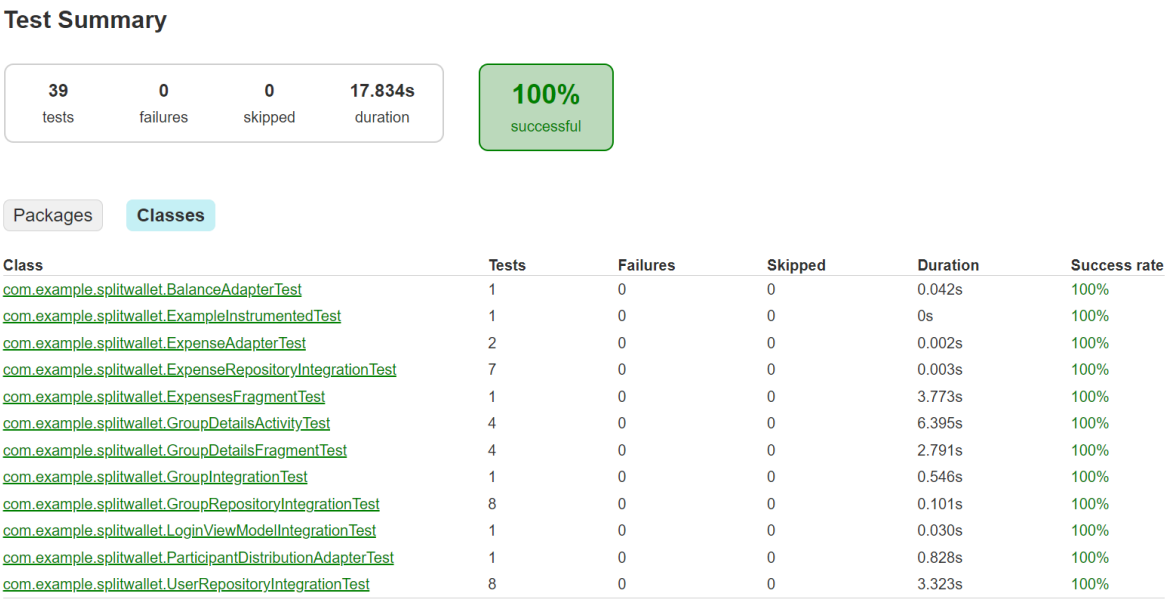


Рисунок 41 Полученная статистика по выполнению интеграционных тестов по классам

Среднее время выполнения тестов: 0,457282.

Рассмотрим наглядное соотношение количества тестов для разных классов.

Рисунок 42 Соотношение количества интеграционных тестов по классам

Можем наблюдать, что большая часть тестов была написана для классов типа Repository, так как именно они взаимодействуют с большей частью компонентов. Для Adapter и Integration было написано меньше всего, так как в них, преимущественно, происходит взаимодействие с одним-двумя классами.

Рассмотрим также соотношение времени выполнения тестов относительно классов.

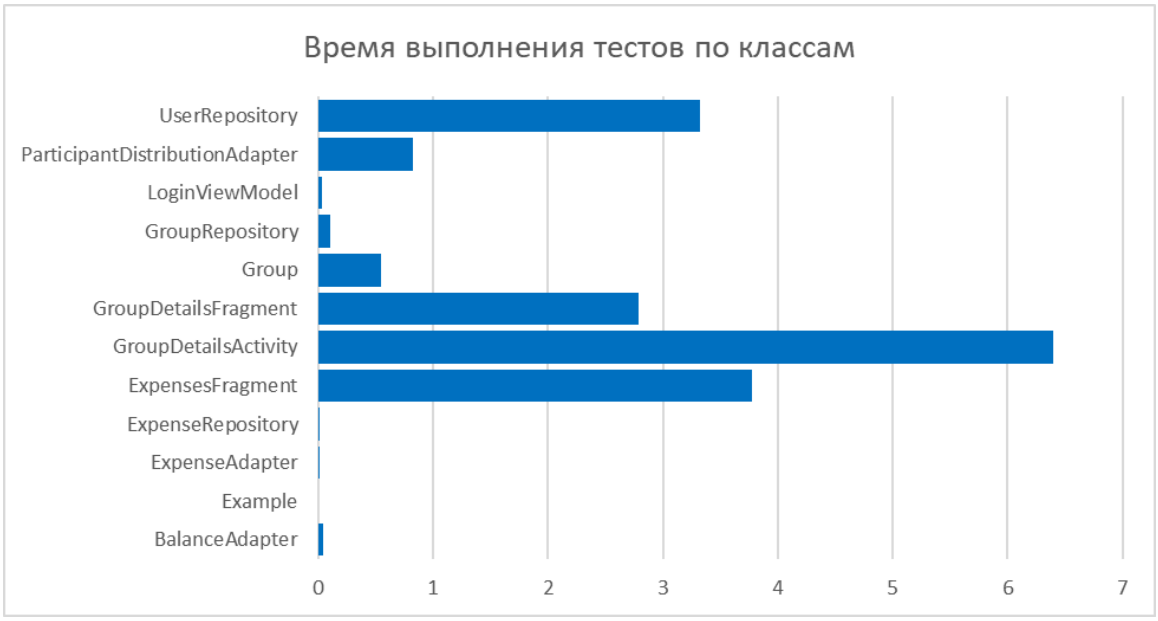


Рисунок 43 Соотношение времени выполнения интеграционных тестов по классам

Однако на этой диаграмме плохо видны (или не видны) меньшие значения из-за большого разброса показателей. Поэтому иллюстрируем динамику изменения времени выполнения в зависимости от класса другим способом.

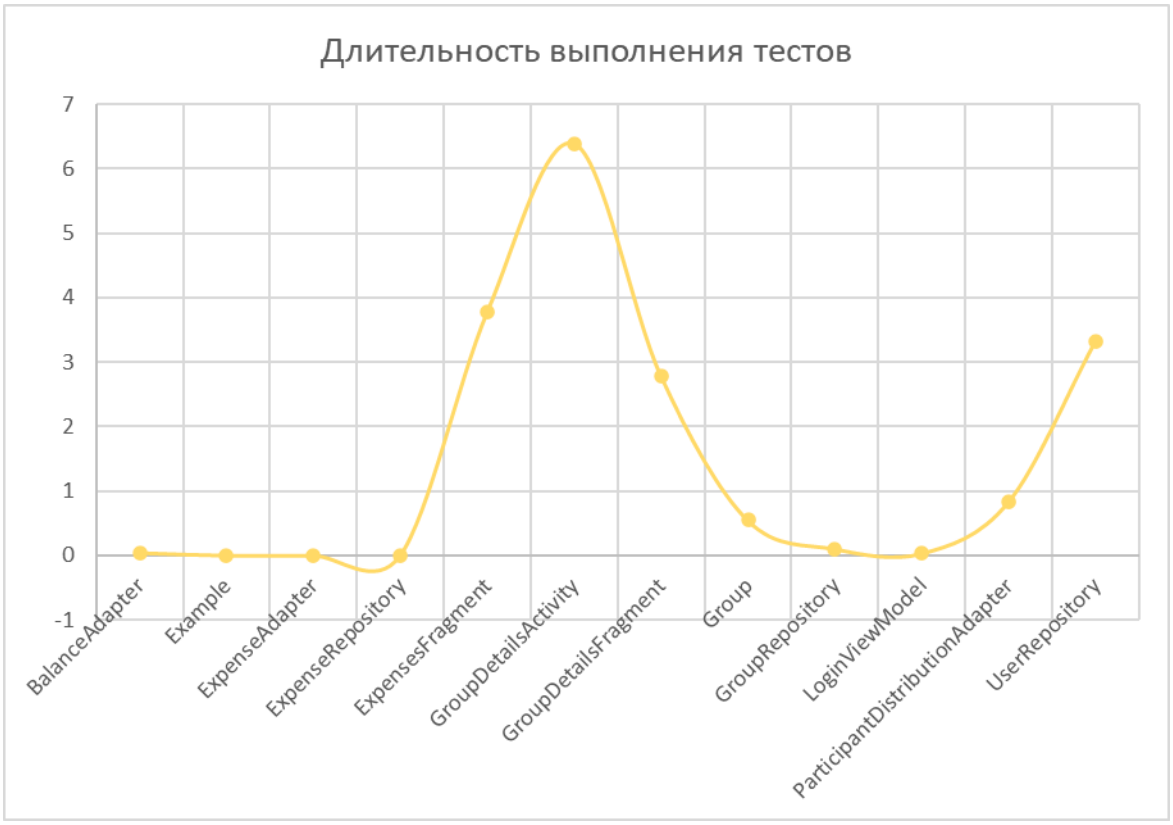


Рисунок 44 График зависимости времени выполнения интеграционных тестов от класса

Пользовательские сценарии (и их составляющие) и тесты, которыми они были реализованы:

- testAddExpenseDialogShownOnFabClick в ExpensesFragmentTest – тестирует нажатие кнопки добавления расхода и открытие окна с ручным вводом данных о нём;

- тесты в классе GroupDetailsActivityTest реализуют пользовательский сценарий: открыть страницу группы, перейти на вкладку со сведениями об участниках, нажать на строку одного участника и посмотреть информацию о нём, выйти из аккаунта;

- GroupDetailsFragmentTest – тестирует разные виды аутентификации пользователя и корректное отображение вкладки “Details” страницы группы;

- GroupIntegrationTest реализует пользовательский сценарий: создание группы одним пользователем, получение им кода для вступления, ввод кода другим пользователем, добавление другого пользователя в группу;

- ParticipantDistributionAdapterTest реализует пользовательский сценарий: создание расхода пользователем, добавление к расходу участника, разделение его суммы между участниками.

Нагрузочное тестирование

Было проведено нагрузочное тестирование с помощью инструмента Apache JMeter для следующих сервисов и эндпоинтов соответственно:

-GET group-service/groups/my : 10000 пользователей ; Ramp-Up Period = 3 (3333 пользователя в секунду)

-GET expenses-service/groups/{groupId}/expenses : 10000 пользователей ; Ramp-Up Period = 3 (3333 пользователя в секунду)

-GET expensesuser-service/groups/{groupId}/expenses/{expenseId} : 10000 пользователей ; Ramp-Up Period = 3 (3333 пользователя в секунду)

-POST auth-service/login: 10000 пользователей ; Ramp-Up Period = 50 (200 пользователей в секунду)

Рассмотрим результаты тестирования сервиса аутентификации (logAuthServuce). Для этого сервиса тестировался эндпоинт POST auth-service/login, т. е. вход пользователя количеством 10000 пользователей и по 200 пользователей в секунду.

Полученные общие результаты по запросам

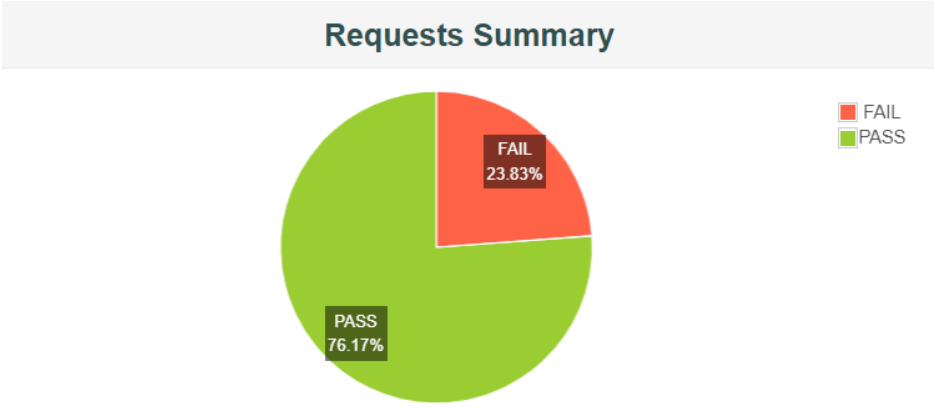


Рисунок 45 Диаграмма соотношения результатов запросов для сервиса logAuthServuce

Из всех заявленных 10000 запросов было корректно обработано 7617 (76.17%), а при остальных 2383 запросах (23.83%) произошли ошибки.

Рассмотрим результаты подробнее.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**10000**	**2383**	**23.83%**	**20279.04**	**1**	**38468**	**22493.00**	**35969.90**	**37154.80**	**38085.98**	**119.62**	**190.43**	**22.90**
HTTP Request	10000	2383	23.83%	20279.04	1	38468	22493.00	35969.90	37154.80	38085.98	119.62	190.43	22.90

Рисунок 46 Общая статистика запросов для сервиса logAuthServuce

Также рассмотрим наглядное представление статистики на графике:

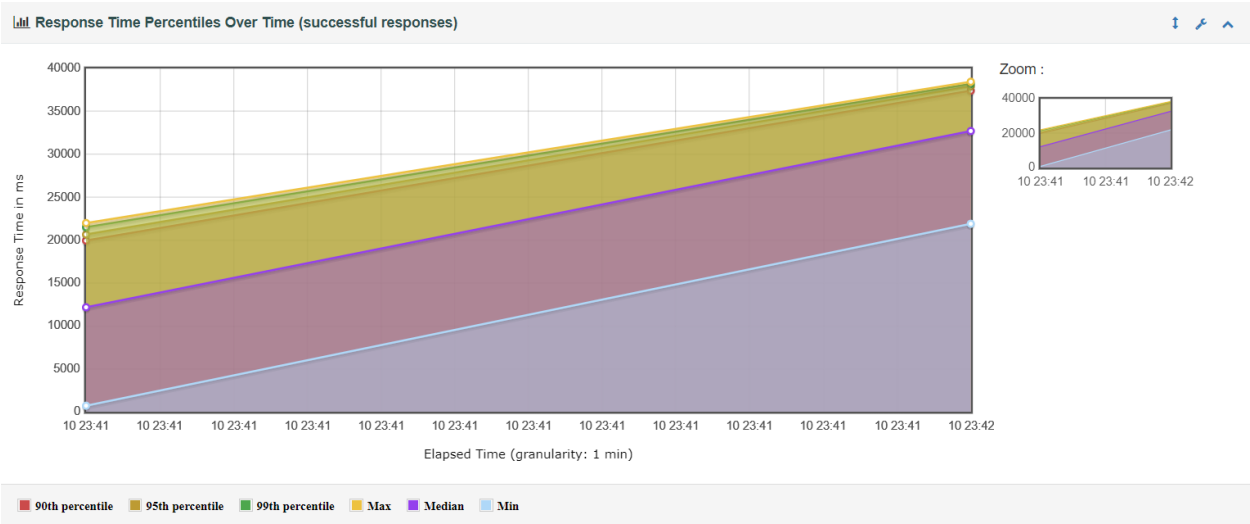


Рисунок 47 График общей статистики запросов для сервиса logAuthServuce

По таблице можно заметить, что:

- Среднее время отклика 20 279.04 – высокое;

- Минимальное время отклика – 1. Вероятно, это из-за кэшированных или быстрых ответов;

- Максимальное время отклика - 38 468. Почти 40 секунд, а это уже критически медленно;

- Небольшое стандартное отклонение- 119.62. Указывает на сохранение такого характера производительности.

Также проанализируем таблицу со статистикой ошибок.

Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: java.net.BindException/Non HTTP response message: Address already in use: connect	1438	60.34%	14.38%
400/Bad Request	945	39.66%	9.45%

Рисунок 48 Статистика ошибок при запросах для сервиса logAuthServuce

Общее количество ошибок: 2383

- BindException: Address already in use – 1438 - 60.34%. Проблема на клиентской стороне JMeter — слишком много потоков или параллельных подключений.

- 400 Bad Request – 945 - 39.66%. Ошибки сервиса — возможно, некорректный формат запроса или данные.

Соответствующие выводы:

1. Низкий процент успешных запросов (76.17%) указывает на неустойчивость сервиса при нагрузке.

2. Время отклика крайне высокое — среднее более 20 секунд, а 90% запросов не укладываются даже в 38 секунд.

3. Ошибка BindException означает, что JMeter перегружает систему, и система не может установить новые соединения (нужно уменьшить количество одновременных потоков или использовать другие порты).

4. 400 Bad Request — это ошибка от сервиса. Следует проверить корректность данных, которые отправлялись при тестировании.

Теперь сравним полученные показатели с ожидаемыми.

Факт: низкий успех теста (76.17%) и высокая задержка (20+ секунд)

Ожидание:

- Число запросов в секунду (RPS) на создание группы, добавление расходов и экспорт данных суммарно превышает 60 RPS.

- Среднее время отклика, для соответствия APDEX ≥ 0.85, должно быть < 500 мс (по целевому порогу).

Реальность:

- Успешность низкая — почти 24% запросов завершились ошибкой.

- Среднее время отклика — ~20 секунд, перцентиль 90 — 38 секунд, что на два порядка хуже ожиданий.

Ошибки соединения (BindException: Address already in use). Это ошибка на стороне JMeter, означающая, что система исчерпала доступные исходящие порты.

Ошибки 400 Bad Request (945 штук). Это реальные ошибки сервиса, т.е. AUTHSERVICE не принял запрос (например, неверный формат данных). Вероятнее всего, JMeter отправлял невалидные параметры.

Ожидание по требованиям: 400 ошибок должно быть минимально, особенно при стандартных сценариях (создание группы, расходы и т.п.).

На основе анализа полученных данных можно составить следующие рекомендации по улучшению в перспективе:

1. Оптимизация обработки входящих запросов

Снизить среднее и пиковое время отклика за счёт:

- устранения "узких мест" в логике аутентификации/авторизации;

- минимизации блокирующих операций (например, обращений к внешним сервисам).

2. Повышение устойчивости к пиковой нагрузке

- внедрить автоматическое масштабирование (autoscaling) для экземпляров сервиса на уровне контейнеров;

- расширить пул соединений и провести настройку пула БД (connection pool), чтобы исключить задержки при высоком RPS;

- настроить rate limiting и throttling, чтобы избежать перегрузки в случае аномального поведения клиентов.

3. Повышение стабильности инфраструктуры тестирования

Перенастроить JMeter:

- использовать дополнительные IP-адреса или балансировку нагрузки;

- ограничить число потоков или ввести паузы между запросами, чтобы избежать BindException,

- обеспечить отправку валидных запросов во избежание 400 Bad Request.

4. Реализация кэширования и CDN

Внедрить кэширование на уровне:

- частых ответов аутентификации (например, валидированные токены);

- статических данных и публичных ключей.

Обеспечить горизонтальное масштабирование компонентов, особенно тех, что обрабатывают регистрацию и вход.

Рассмотрим результаты тестирования сервиса по управлению расходами (logExpensesService). Для этого сервиса тестировался эндпоинт GET expenses-service/groups/{groupId}/expenses, т. е. получение пользователем всех расходов определённой группы количеством 10000 пользователей и по 3333 пользователей в секунду.

Полученные общие результаты по запросам:

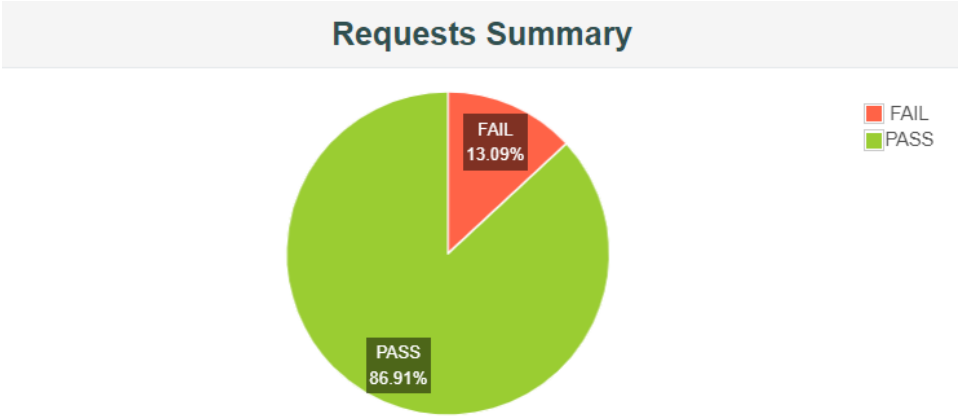


Рисунок 49 Диаграмма соотношения результатов запросов для сервиса logExpensesService

Из всех заявленных 10000 запросов было корректно обработано 8691 (86.91%), а при остальных 1309 запросах (13.09%) произошли ошибки.

Рассмотрим результаты подробнее.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**10000**	**1309**	**13.09%**	**5950.35**	**0**	**12823**	**6847.50**	**7541.90**	**8813.90**	**11707.89**	**695.07**	**991.01**	**829.44**
HTTP Request	10000	1309	13.09%	5950.35	0	12823	6847.50	7541.90	8813.90	11707.89	695.07	991.01	829.44

Рисунок 50 Общая статистика запросов для сервиса logExpensesService

Также рассмотрим наглядное представление статистики на графике:

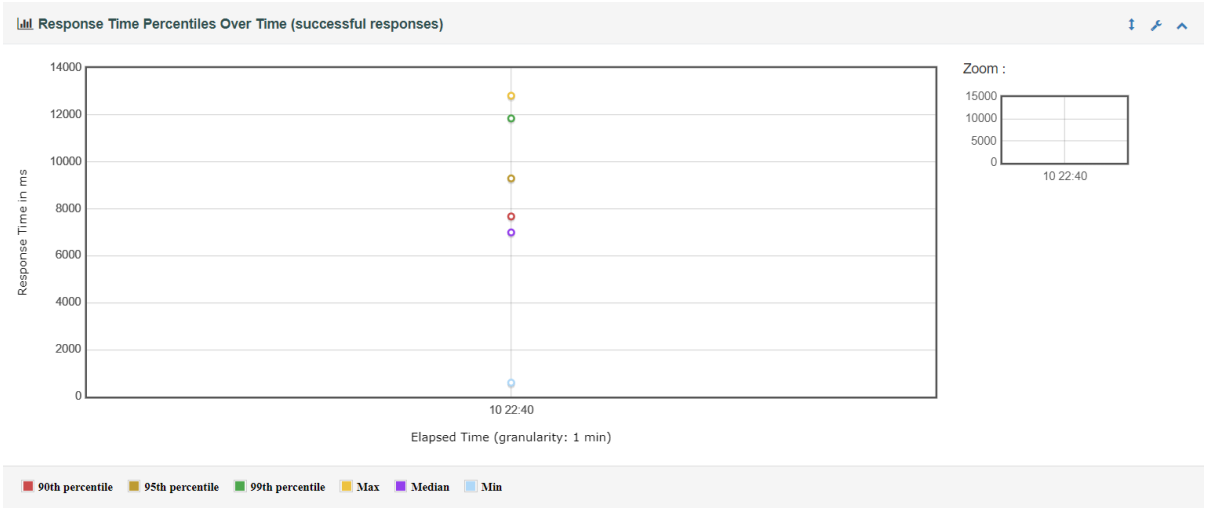


Рисунок 51 График общей статистики запросов для сервиса logExpensesService

По таблице можно заметить, что:

- Среднее время отклика - 5 950.35 – высокое;

- Минимальное время отклика – 0. Скорее всего, мгновенный отказ или кэшированный ответ;

- Максимальное время отклика - 12 823. Почти 13 секунд — долго для обычных операций;

- Стандартное отклонение- -991.01. Значительная вариативность.

Также проанализируем таблицу со статистикой ошибок

Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: java.net.BindException/Non HTTP response message: Address already in use: connect	1241	94.81%	12.41%
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to localhost:8080 [localhost/127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed: Connection refused: connect	68	5.19%	0.68%

Рисунок 52 Статистика ошибок при запросах для сервиса logExpensesService

Общее количество ошибок: 1 309

- BindException: Address already in use - 1 241 - 94.81%. Клиентская ошибка JMeter — слишком много потоков или исчерпаны порты.

- HttpHostConnectException: Connection refused – 68 - 5.19%. Сервис был недоступен — не запущен или сбой в момент теста.

Соответствующие выводы:

1. 13.09% сбоев — это много. Сервис не выдерживает даже моделируемую нагрузку. APDEX 0.004 означает крайне неудовлетворительную производительность: почти все пользователи получают медленные ответы.

2. Основная ошибка — BindException, что указывает на проблемы с самим тестированием (JMeter не может открывать соединения).

3. 68 ошибок Connection refused говорят о том, что сервис мог упасть или не быть доступным на момент теста.

Теперь сравним полученные показатели с ожидаемыми.

Факт: средняя успешность теста (86.91%) и высокая задержка (5–12 секунд)

Ожидание:

- Суммарная нагрузка по операциям (добавление расходов, экспорт) — более 45 RPS.

- Среднее время отклика — < 500 мс, чтобы обеспечить APDEX ≥ 0.85.

Реальность:

- Успешность ниже 90% — каждый восьмой запрос завершился ошибкой.

- Среднее время отклика ~5 950 мс, 90-й перцентиль почти 12 секунд — в 10–25 раз хуже допустимого уровня.

Ошибки соединения (BindException: Address already in use — 1 241 шт). Ошибка на стороне JMeter, когда клиент не может открыть новые соединения из-за исчерпания портов.

Ошибки подключения (Connection refused — 68 шт). Сервис не был доступен в момент части теста: либо был не запущен, либо перегружен и упал.

Ожидание по требованиям: сервис должен быть стабилен при нагрузке до 30 RPS на добавление расходов и до 17 RPS на экспорт.

На основе анализа полученных данных можно составить следующие рекомендации по улучшению в перспективе:

1. Устранить BindException:

- увеличить диапазон портов в ОС (net.ipv4.ip_local_port_range);

- использовать распределённое тестирование JMeter (несколько агентов).

2. Разобраться с высокой задержкой:

- провести профилирование (например, Java Flight Recorder или JMC)

3. Масштабирование:

- убедиться, что EXPENSESERVICE масштабируется горизонтально.

Рассмотрим результаты тестирования сервиса, отвечающего за управление участниками расходов (logExpenseUserService). Для этого сервиса тестировался эндпоинт GET expensesuser-service/groups/{groupId}/expenses/{expenseId}, т. е. получение пользователем

информации о расходе в конкретной группе количеством 10000 пользователей и по 3333 пользователей в секунду.

Полученные общие результаты по запросам:

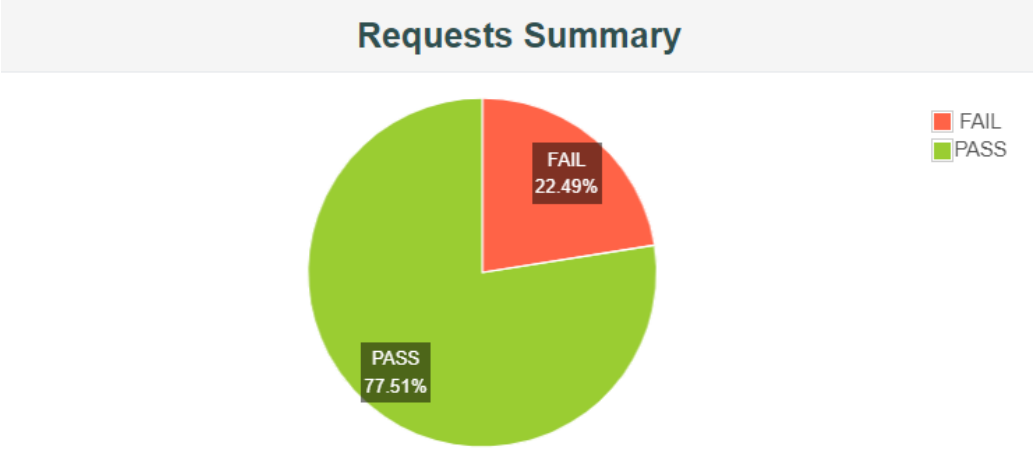


Рисунок 53 Диаграмма соотношения результатов запросов для сервиса logExpenseUserService

Из всех заявленных 10000 запросов было корректно обработано 7751 (77.51%), а при остальных 2249 запросах (22.49%) произошли ошибки.

Рассмотрим результаты подробнее.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	2249	22.49%	4958.70	0	8150	5977.50	7320.90	7499.00	7889.98	868.58	920.45	929.05
HTTP Request	10000	2249	22.49%	4958.70	0	8150	5977.50	7320.90	7499.00	7889.98	868.58	920.45	929.05

Рисунок 54 Общая статистика запросов для сервиса logExpenseUserService

Также рассмотрим наглядное представление статистики на графике:

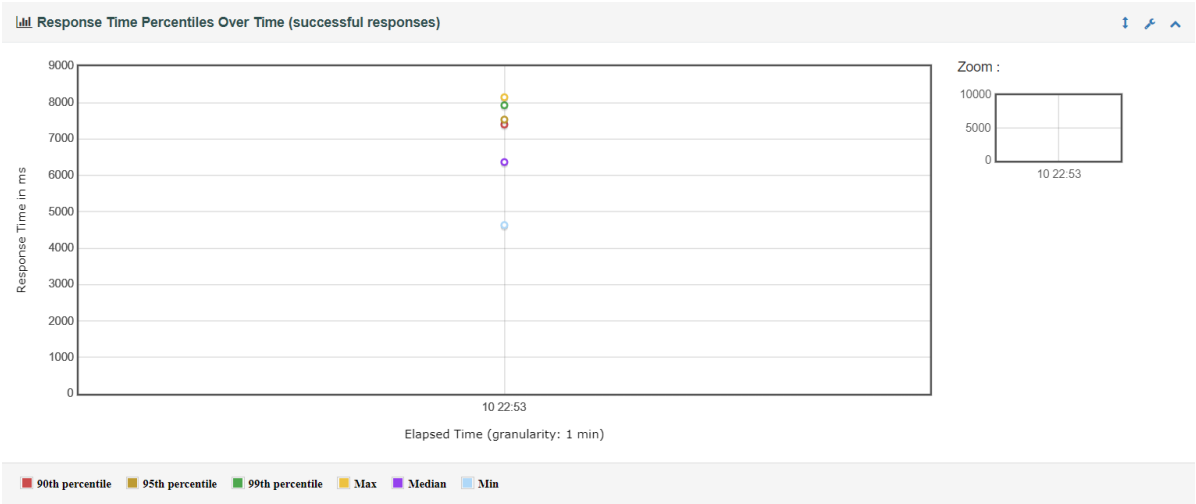


Рисунок 55 График общей статистики запросов для сервиса logExpenseUserService

По таблице можно заметить, что:

- Среднее время отклика - 4 958.70. Почти 5 секунд — в 10 раз превышает целевой порог;

- Минимальное время отклика – 0. Вероятно, кэшированный или мгновенный ответ;

- Максимальное время отклика - 8 150. Значительное — более 8 секунд;

- Стандартное отклонение - 868.58. Значительная вариативность.

Также проанализируем таблицу со статистикой ошибок

Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: java.net.BindException/Non HTTP response message: Address already in use: connect	1110	49.36%	11.10%
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to localhost:8080 [localhost/127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed: Connection refused: connect	872	38.77%	8.72%
Non HTTP response code: java.net.SocketException/Non HTTP response message: No buffer space available (maximum connections reached?): connect	261	11.61%	2.61%
500/Internal Server Error	6	0.27%	0.06%

Рисунок 56 Статистика ошибок при запросах для сервиса logExpenseUserService

Общее количество ошибок: 2 249.

- BindException: Address already in use - 1 110 - 49.36%. JMeter не смог создать соединение — исчерпаны порты;

- HttpHostConnectException: Connection refused – 872 - 38.77%. Сервис не принял соединение — возможно, был недоступен;

- SocketException: No buffer space available – 261 - 11.61%. Недостаточно ресурсов ОС для установления соединения;

- 500 Internal Server Error – 6 - 0.27%. Реальная ошибка на стороне сервиса — нужно проверить логи.

Соответствующие выводы:

1. Процент успешных запросов — 77.51% — недостаточен для стабильной работы в продакшн-среде.

2. Время отклика — от 5 до 8 секунд у большинства запросов — в 10–15 раз выше ожидаемого.

3. Ошибки BindException, SocketException — проблемы конфигурации JMeter: слишком много потоков, нехватка портов и системных ресурсов.

4. Ошибки HttpHostConnectException — возможно, сервис не выдержал нагрузку.

5. Ошибка 500 — указывает на сбой внутри сервиса, пусть и единичный.

Теперь сравним полученные показатели с ожидаемыми.

Факт: низкий процент успешных запросов и высокая задержка

Ожидание:

- Успешность запросов должна составлять ≥ 99%;

- Среднее время отклика должно быть < 500 мс для соответствия APDEX ≥ 0.85;

- Сервис должен обрабатывать до 30–60 RPS по ключевым операциям.

Реальность:

- Успешность — 77.51% (22.49% ошибок);

- Среднее время отклика — ~4 959 мс;

- 90% запросов укладываются в ~7.9 секунд — в 15+ раз хуже допустимого.

Ошибки подключения (BindException, SocketException, Connect Refused). Говорят о некорректной конфигурации JMeter или о том, что сервис не принимал соединения (возможно, падал под нагрузкой). Сервис не масштабирован или не устойчив к пиковым подключениям

Внутренние ошибки сервиса (500 Internal Server Error). Ошибки 5xx должны отсутствовать при корректной нагрузке. Операции создания, редактирования, запроса данных должны работать стабильно. 6 ошибок 500 — единичные, но важные сбои на стороне сервиса.

На основе анализа полученных данных можно составить следующие рекомендации по улучшению в перспективе:

1. Именить конфигурацию JMeter:

- Уменьшить количество потоков;
- Добавить задержки (think time);
- Увеличить диапазон портов и системные ресурсы.

2. Проверить устойчивость сервиса:

- Убедиться, что сервис не падает при резкой нагрузке;
- Анализировать, почему 8080 не принимал соединения;

3. Решить проблему с буферами (SocketException):

- Возможно стоит изменить системные настройки (например, ulimit, net.ipv4.tcp_tw_reuse)

4. Разобраться с 500 Internal Server Error:

- Посмотреть в логах, что вызвало исключение на стороне сервиса.

5. Повторить тестирование с поэтапным увеличением нагрузки:

- Отдельно протестировать каждую операцию;
- Следить за стабильностью на каждом этапе.

Рассмотрим результаты тестирования сервиса, отвечающего за управление группами (logGroupService). Для этого сервиса тестировался

эндпоинт GET group-service/groups/my : 10000 пользователей ; Ramp-Up Period = 3 (3333 пользователя в секунду), т. е. получение пользователем информации о своих группах количеством 10000 пользователей и по 3333 пользователей в секунду.

Полученные общие результаты по запросам:

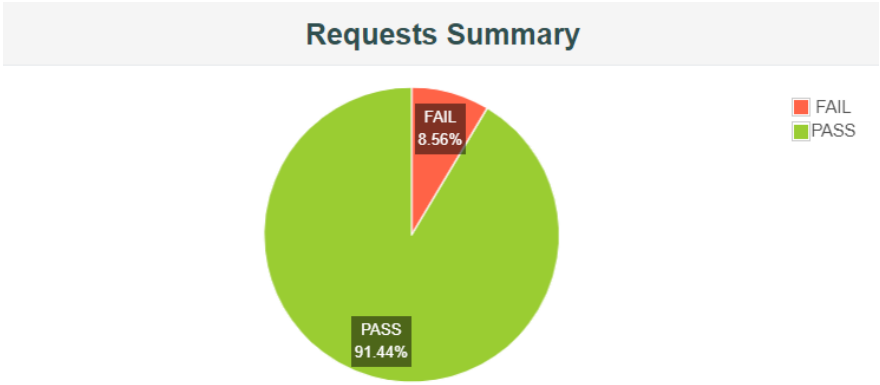


Рисунок 57 Диаграмма соотношения результатов запросов для сервиса logGroupService

Из всех заявленных 10000 запросов было корректно обработано 9144 (91.44%), а при остальных 856 запросах (8.56%) произошли ошибки.

Рассмотрим результаты подробнее.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	856	8.56%	1699.21	0	4787	2039.00	2863.00	3114.00	3647.00	923.36	5167.90	1143.63
HTTP Request	10000	856	8.56%	1699.21	0	4787	2039.00	2863.00	3114.00	3647.00	923.36	5167.90	1143.63

Рисунок 58 Общая статистика запросов для сервиса logGroupService

Также рассмотрим наглядное представление статистики на графике:

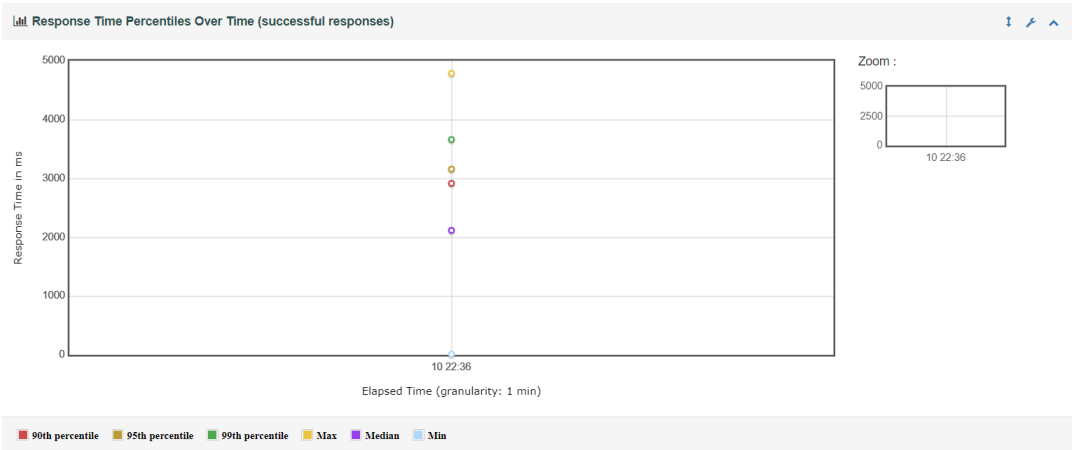


Рисунок 59 График общей статистики запросов для сервиса logGroupService

По таблице можно заметить, что:

- Среднее время отклика - 1699.21. Более 1.6 сек — не подходит для интерактивных операций;

- Минимальное – 0;

- Максимальное – 4787. Почти 5 сек долго;

- Стандартное отклонение - 923.36. Умеренное отклонение — задержки относительно стабильны.

Производительность неудовлетворительная — необходима оптимизация времени отклика.

Также проанализируем таблицу со статистикой ошибок

Errors

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: org.apache.http.conn.HttpHostConnectException/Non HTTP response message: Connect to localhost:8080 [localhost/127.0.0.1, localhost/0:0:0:0:0:0:0:1] failed: Connection refused: connect	856	100.00%	8.56%

Рисунок 60 Статистика ошибок при запросах для сервиса logGroupService

Общее количество ошибок: 856

- ConnectException: Connection refused – 856 - 100.00%. Сервис не принимал соединения — упал под нагрузкой.

Соответствующие выводы:

1. Доля ошибок (8.56%) выше нормы — это означает, что сервис некорректно справляется с нагрузкой;

2. Все ошибки — Connection refused, что указывает либо на недоступность сервиса, либо на то, что он не выдерживает входящий трафик;

3. Среднее время отклика в 1.7 сек — уже превышает допустимые пороги для интерактивных операций.

Теперь сравним полученные показатели с ожидаемыми.

Факт: ошибки подключения и высокая задержка

Ожидание:

- Успешность запросов — ≥ 99%;

- Время отклика — < 500 мс, чтобы соответствовать APDEX ≥ 0.85;

- Срвис управления группами должен справляться с высокой нагрузкой, особенно при массовом создании и доступе к группам.

Реальность:

- Успешность — 91.44%, то есть 8.56% ошибок;

- Среднее время отклика — ~1 699 мс, что в 3 раза выше допустимого;

- APDEX — 0.206, значительно ниже нормы.

Ошибки подключения (Connect Refused), т.е. сервис не принимал соединения. Это либо результат краша/перегрузки GROUPSERVICE, либо он не был запущен/готов во время теста. Проблема может быть как на стороне сервиса, так и в JMeter-сценарии (например, преждевременный старт).

На основе анализа полученных данных можно составить следующие рекомендации по улучшению в перспективе:

1. Анализ логов и инфраструктуры:

Есть ли краши, рестарты, ошибки запуска? Был ли сервис перегружен?

2. Профилирование задержек:

Замерить время обработки по маршрутам, локализовать "узкие места".

3. Повторное тестирование:

После устранения проблем провести тест повторно с той же нагрузкой для верификации улучшений.

Выводы

В процессе выполнения курсовой работы было реализовано мобильное приложение, позволяющее решить проблему разделения расходов между пользователями. Были проведены расчёты предполагаемых соотношения R/W нагрузки и объемов трафика и объемов дисковой системы. Были проведены юнит- интеграционное и нагрузочное тестирования. Результаты первых двух видов показали, что основной заявленный функционал приложения работает корректно, однако при нагрузочном тестировании выявились некоторые ошибки. Их исправление можно оставить для будущего развитие проекта.