



# Jak optymalnie korzystać z systemu kontroli wersji

28.04.2022

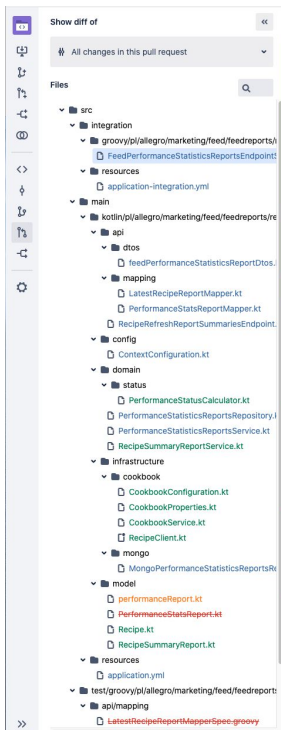
**allegro**

# Agenda

Definicja problemu	5"
Poszukiwanie rozwiązania	15"
Podsumowanie	5"

# Definicja problemu

# Przykładowy PR



src / main / kotlin / pl / allegro / marketing / feed / feedreports / reports / infrastructure / mongo / MongoPerformanceStatisticsReportsRepository.kt MODIFIED

```

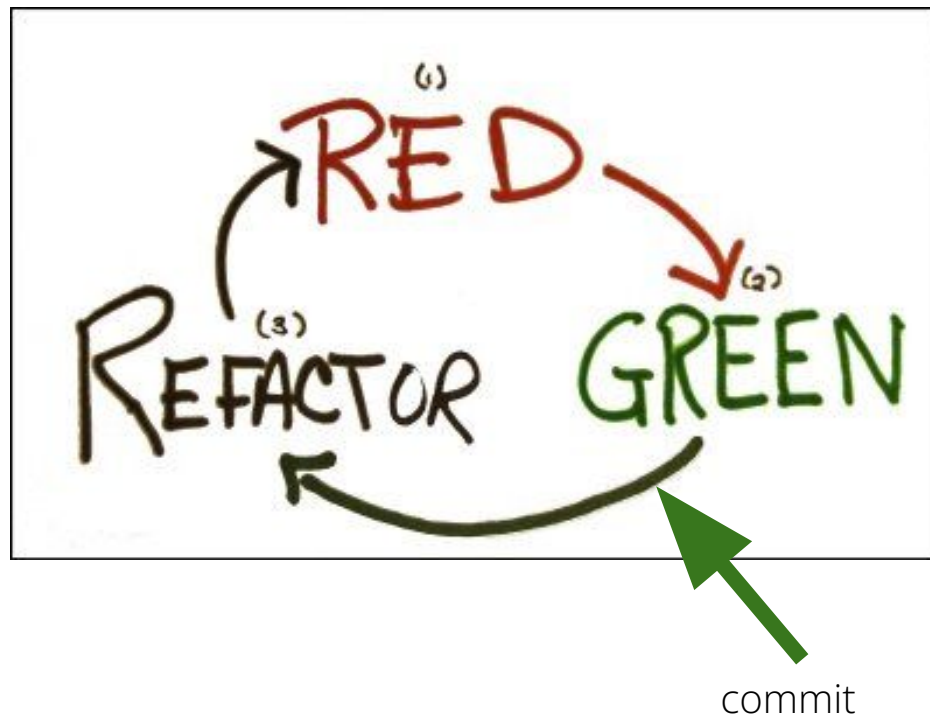
4 + import org.springframework.data.mongodb.core.aggregation.Aggregation
5 - import org.springframework.data.mongodb.core.query.Criteria.where
6 - import org.springframework.data.mongodb.core.query.Query.query
7 + import org.springframework.data.mongodb.core.query.InValues
8 - import pl.allegro.marketing.feed.feedreports.reports.domain.PerformanceStatisticsReportsRepository
9 + import pl.allegro.marketing.feed.feedreports.reports.model.AveragePerformanceStatsReport
10 - import pl.allegro.marketing.feed.feedreports.reports.model.PerformanceStatsReport
11 - import pl.allegro.marketing.feed.feedreports.reports.model.RecipeId
12 - import pl.allegro.tech.common.andamio.metrics.micrometer.Timed
13 + import java.math.BigDecimal
14 - import java.time.Instant
15 - import java.time.LocalDate
16 - import java.time.LocalDateTime
17 - import java.time.ZoneOffset.UTC
18 -
19 - class MongoPerformanceStatisticsReportsRepository(private val mongoOperations: MongoOperations) : PerformanceStatisticsReportsRepository {
20 + class MongoPerformanceStatisticsReportsRepository(private val mongoOperations: MongoOperations) {
21 +     PerformanceStatisticsReportsRepository {
22 +
23 +         @Timed(name = "mongodb.statsReports.save")
24 +         override fun save(report: PerformanceStatsReport): PerformanceStatsReport =
25 +             mongoOperations.save(report.toEntity()).toDomain()
26 +
27 +         @Timed(name = "mongodb.statsReports.fetchAllByidAndTime")
28 +         override fun fetch(recipeId: RecipeId, date: LocalDate): PerformanceStatsReport =
29 +             mongoOperations.findOne(
30 +                 query(where("recipeId").isEqualTo(recipeId)
31 +                     .and("date").isEqualTo(date.toInstant())),
32 +                 PerformanceStatsReportEntity::class.java)?.toDomain()
33 +             ?: throw PerformanceStatisticsReportNotFoundException(recipeId, date)
34 +
35 +         @Timed(name = "mongodb.statsReports.fetchRecipesByidAndTime")
36 +         override fun fetch(
37 +             recipeIds: List<RecipeId>,
38 +             startDate: LocalDate,
39 +             endDate: LocalDate
40 +         ): List<AveragePerformanceStatsReport> =
41 +             mongoOperations.aggregate(
42 +                 newAggregation(
43 +                     match(
44 +                         where("id.recipeId".inValues(recipeIds)
45 +                             .and("id.date").gte(startDate.toInstant()).lte(endDate.toInstant()))
46 +                     ),
47 +                     group("id.recipeId", "as"("recipeId")
48 +                         .average("views")
49 +                         .average("viewsMobile")
50 +                         .average("buys")
51 +                         .average("buysMobile")
52 +                         .average("gmw")
53 +                         .average("gmwMobile")
54 +                     )
55 +                 )
56 +             )

```

- Wiele zmienionych plików
- Wiele logicznych zmian w pojedynczym pliku
- Brak informacji "gdzie" zacząć code review
- Utrudnione znalezienie potencjalnych błędów lub rozbieżności z założeniami
- Szczególne utrudnienie dla nowych pracowników



## Cykl TDD

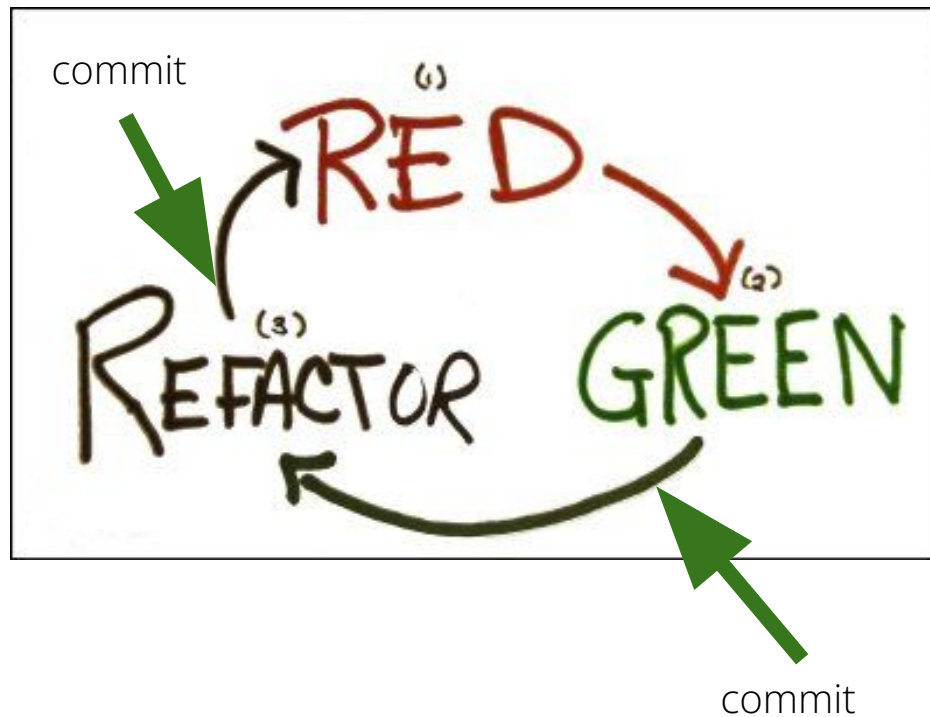


# Cykl TDD

The screenshot illustrates the TDD (Test-Driven Development) cycle in an IDE. The top part shows the 'Commit Changes' dialog, indicating that the file `MongoPerformanceStatisticsReportsRepository.kt` has been modified. The commit message is `GUZIEC-2222 Optimize aggregation in Mongo on fetching recipes`. The bottom part shows the 'Run' window, displaying the test results: `Tests passed: 175 of 175 tests - 34 sec 574 ms`. The log output shows the application shutting down successfully.

- commitujemy tylko te zmiany, które:
  - **kompilują się**
  - **przechodzą wszystkie testy**

# Cykl TDD





## Zmiana nazwy klasy

- commitujemy tylko te zmiany, które:
  - kompilują się
  - przechodzą wszystkie testy
- separujemy niezależne zmiany:
  - **zmiana nazwy klasy/funkcji/zmiennej**

## Podbicie wersji zależności

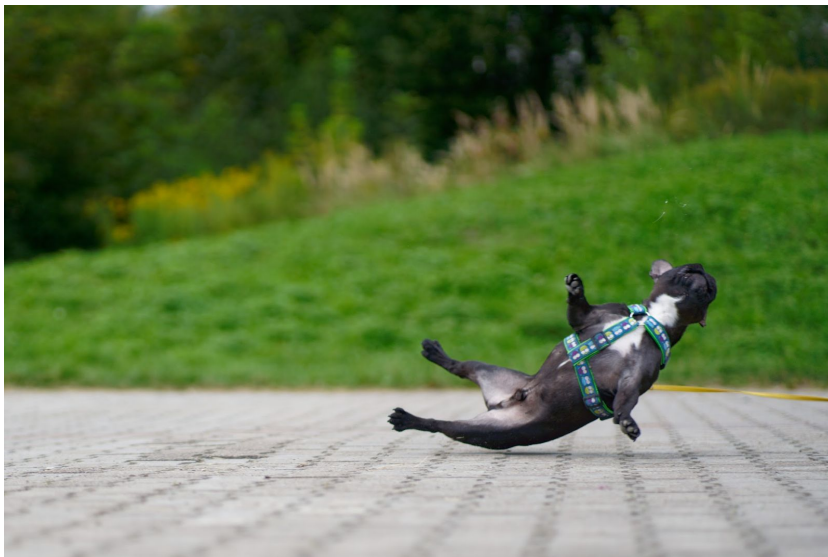
- commitujemy tylko te zmiany, które:
  - kompilują się
  - przechodzą wszystkie testy
- separujemy niezależne zmiany:
  - zmiana nazwy klasy/funkcji/zmiennnej
  - **podbicie wersji zależności**

# Separujemy niezależne zmiany



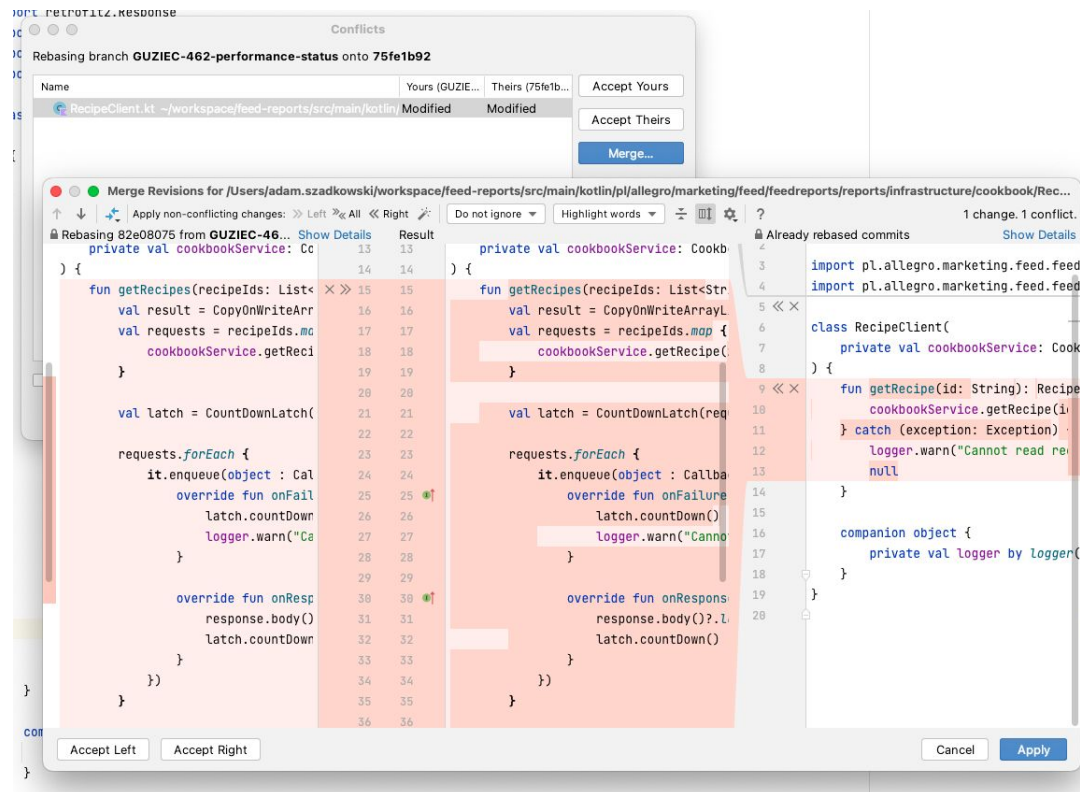
- commitujemy tylko te zmiany, które:
  - kompilują się
  - przechodzą wszystkie testy
- separujemy niezależne zmiany:
  - zmiana nazwy klasy/funkcji/zmiennnej
  - podbicie wersji zależności
  - **przenoszenie pliku między folderami**
  - **przenoszenie klasy między package'ami**
  - **zmiana formatowania pliku**

# Pierwsze wątpliwości

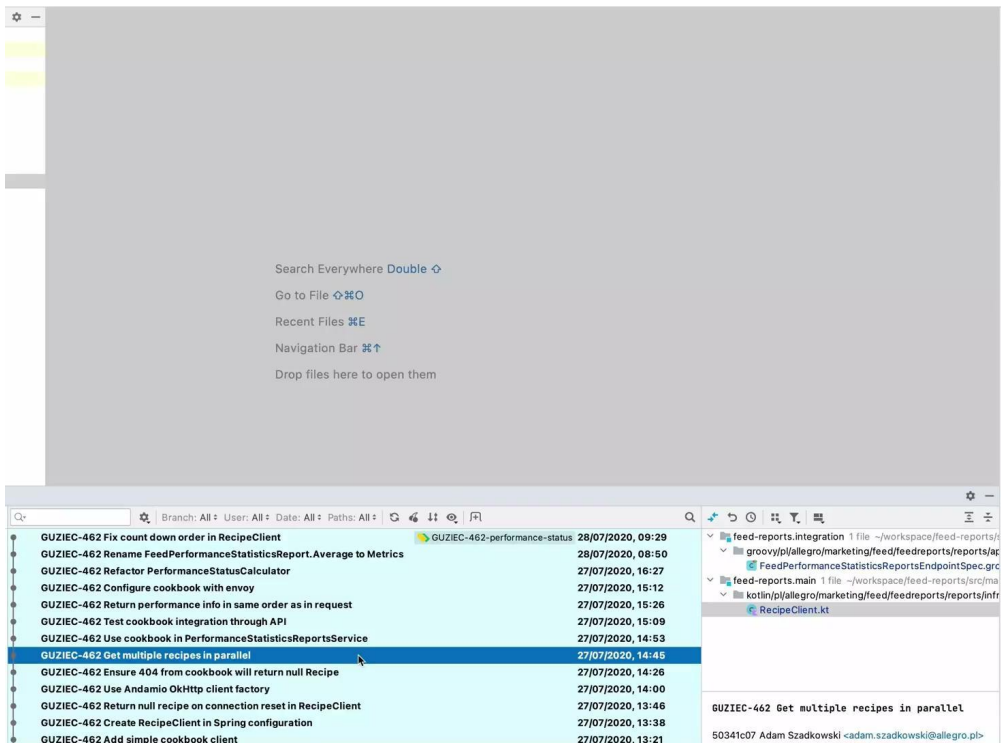


- to przecież wymaga dużo więcej pracy
- to się łączy tylko z TDD
- co zrobić, kiedy się pomylimy
- jak sobie poradzić kiedy piszę nowy kod i muszę np. przenieść klasę

# Rebase



# Przepisywanie historii

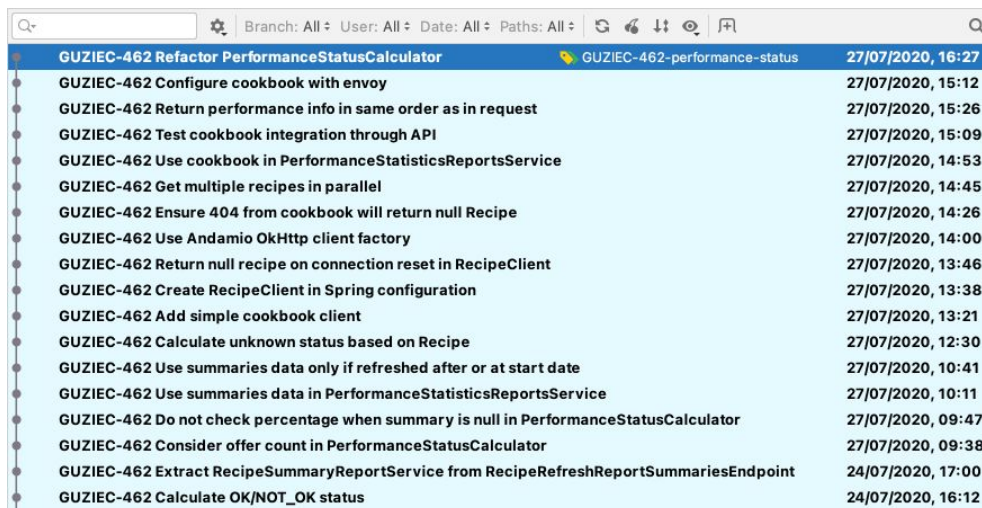


- możemy tylko te relacje klasy: przesunąć "fixup" w odpowiednią się miejsce w historii
  - przekształca się w commit jako "edit"
- możemy nie dodawać ostatniego commita
  - **możemy dodać nowy commit**
    - podbicie wersji zależności
    - przenoszenie pliku między folderami
    - przenoszenie klasy między package'ami
    - zmiana formatowania pliku
- przepisywanie historii w gicie jest ok (dopóki nie wystawimy PR)
  - **zamiast dodawać kolejny commit albo dorzucać poprawkę do innego - można dopisać do oryginalnego commita**

# Odkładanie zmian na półkę

- commitujemy tylko te zmiany, które:
  - kompilują się
  - przechodzą wszystkie testy
- separujemy niezależne zmiany:
  - zmiana nazwy klasy/funkcji/zmiennej
  - podbicie wersji zależności
  - przenoszenie pliku między folderami
  - przenoszenie klasy między package'ami
  - zmiana formatowania pliku
- przepisywanie historii w gicie jest ok (dopóki nie wystawimy PR)
  - zamiast dodawać kolejny commit albo dorzucać poprawkę do innego - można dopisać do oryginalnego commita
- grupujemy zmiany w logiczne "paczki"
  - **zmiany w jednym commicie powinny być ze sobą ściśle powiązane**

# Nazwy commitów

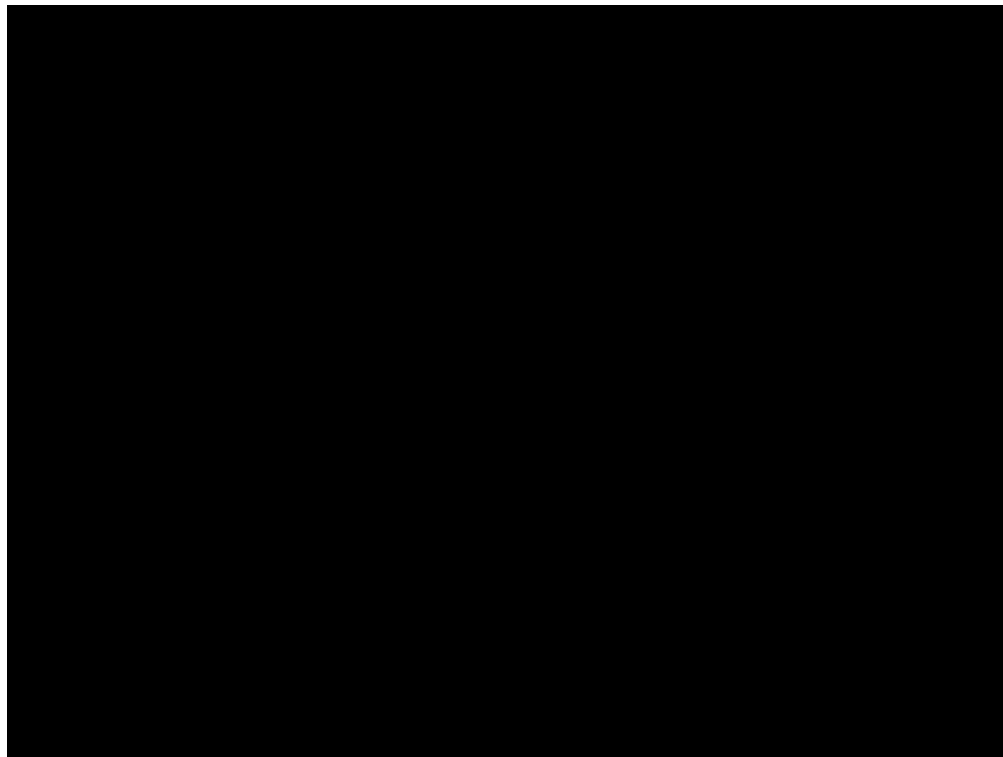


Commit Message	Commit Hash	Timestamp
GUZIEC-462 Refactor PerformanceStatusCalculator	GUZIEC-462-performance-status	27/07/2020, 16:27
GUZIEC-462 Configure cookbook with envoy		27/07/2020, 15:12
GUZIEC-462 Return performance info in same order as in request		27/07/2020, 15:26
GUZIEC-462 Test cookbook integration through API		27/07/2020, 15:09
GUZIEC-462 Use cookbook in PerformanceStatisticsReportsService		27/07/2020, 14:53
GUZIEC-462 Get multiple recipes in parallel		27/07/2020, 14:45
GUZIEC-462 Ensure 404 from cookbook will return null Recipe		27/07/2020, 14:26
GUZIEC-462 Use Andamio OkHttp client factory		27/07/2020, 14:00
GUZIEC-462 Return null recipe on connection reset in RecipeClient		27/07/2020, 13:46
GUZIEC-462 Create RecipeClient in Spring configuration		27/07/2020, 13:38
GUZIEC-462 Add simple cookbook client		27/07/2020, 13:21
GUZIEC-462 Calculate unknown status based on Recipe		27/07/2020, 12:30
GUZIEC-462 Use summaries data only if refreshed after or at start date		27/07/2020, 10:41
GUZIEC-462 Use summaries data in PerformanceStatisticsReportsService		27/07/2020, 10:11
GUZIEC-462 Do not check percentage when summary is null in PerformanceStatusCalculator		27/07/2020, 09:47
GUZIEC-462 Consider offer count in PerformanceStatusCalculator		27/07/2020, 09:38
GUZIEC-462 Extract RecipeSummaryReportService from RecipeRefreshReportSummariesEndpoint		24/07/2020, 17:00
GUZIEC-462 Calculate OK/NOT_OK status		24/07/2020, 16:12

- commitujemy tylko te zmiany, które:
  - kompilują się
  - przechodzą wszystkie testy
- separujemy niezależne zmiany:
  - zmiana nazwy klasy/funkcji/zmiennej
  - podbicie wersji zależności
  - przenoszenie pliku między folderami
  - przenoszenie klasy między package'ami
  - zmiana formatowania pliku
- przepisywanie historii w gicie jest ok (dopóki nie wystawimy PR)
  - zamiast dodawać kolejny commit albo dorzucać poprawkę do innego - można dopisać do oryginalnego commita
- grupujemy zmiany w logiczne "paczki"
  - zmiany w jednym commicie powinny być ze sobą ściśle powiązane
- **nazwa commita powinna dokumentować wprowadzone zmiany**



# Zmiany w obrębie jednego pliku



- commitujemy tylko te zmiany, które:
  - kompilują się
  - przechodzą wszystkie testy
- separujemy niezależne zmiany:
  - zmiana nazwy klasy/funkcji/zmiennej
  - podbicie wersji zależności
  - przenoszenie pliku między folderami
  - przenoszenie klasy między package'ami
  - zmiana formatowania pliku
- przepisywanie historii w gicie jest ok (dopóki nie wystawimy PR)
  - zamiast dodawać kolejny commit albo dorzucać poprawkę do innego - można dopisać do oryginalnego commita
- grupujemy zmiany w logiczne "paczki"
  - zmiany w jednym commicie powinny być ze sobą ściśle powiązane
- nazwa commita powinna dokumentować wprowadzone zmiany

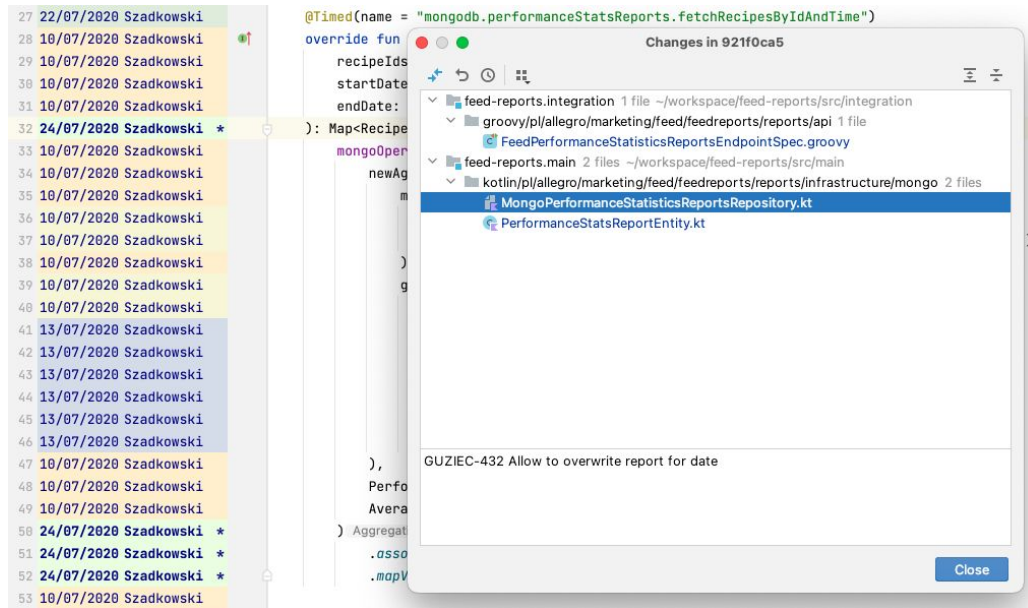
# Podsumowanie

# Zbiór zasad odnośnie commitowania

- commitujemy tylko te zmiany, które:
  - kompilują się
  - przechodzą wszystkie testy
- separujemy niezależne zmiany:
  - zmiana nazwy klasy/funkcji/zmiennnej
  - podbicie wersji zależności
  - przenoszenie pliku między folderami
  - przenoszenie klasy między package'ami
  - zmiana formatowania pliku
- przepisywanie historii w gicie jest ok (dopóki nie wystawimy PR)
  - zamiast dodawać kolejny commit albo dorzucać poprawkę do innego - można dopisać do oryginalnego commita
- grupujemy zmiany w logiczne "paczki"
  - zmiany w jednym commicie powinny być ze sobą ściśle powiązane
- nazwa commita powinna dokumentować wprowadzone zmiany

# Korzyści

- dla twórców:
  - uporządkowanie tego co jest do zrobienia (jakie zmiany kolejno musimy wprowadzić)
  - utrzymywanie cały czas porządku w kodzie
- dla reviewerów:
  - małe zmiany proste w przeglądaniu
  - nazwy commitów, które dobrze opisują co zostało zmienione
  - historia commitów przedstawia proces
  - można robić review "na raty"
- dla wszystkich:
  - każda linijka kodu ma swój komentarz, który pokazuje wszystkie powiązane zmiany
  - łatwiej osiągnąć continuous delivery





Dziękuję!

**allegro**

