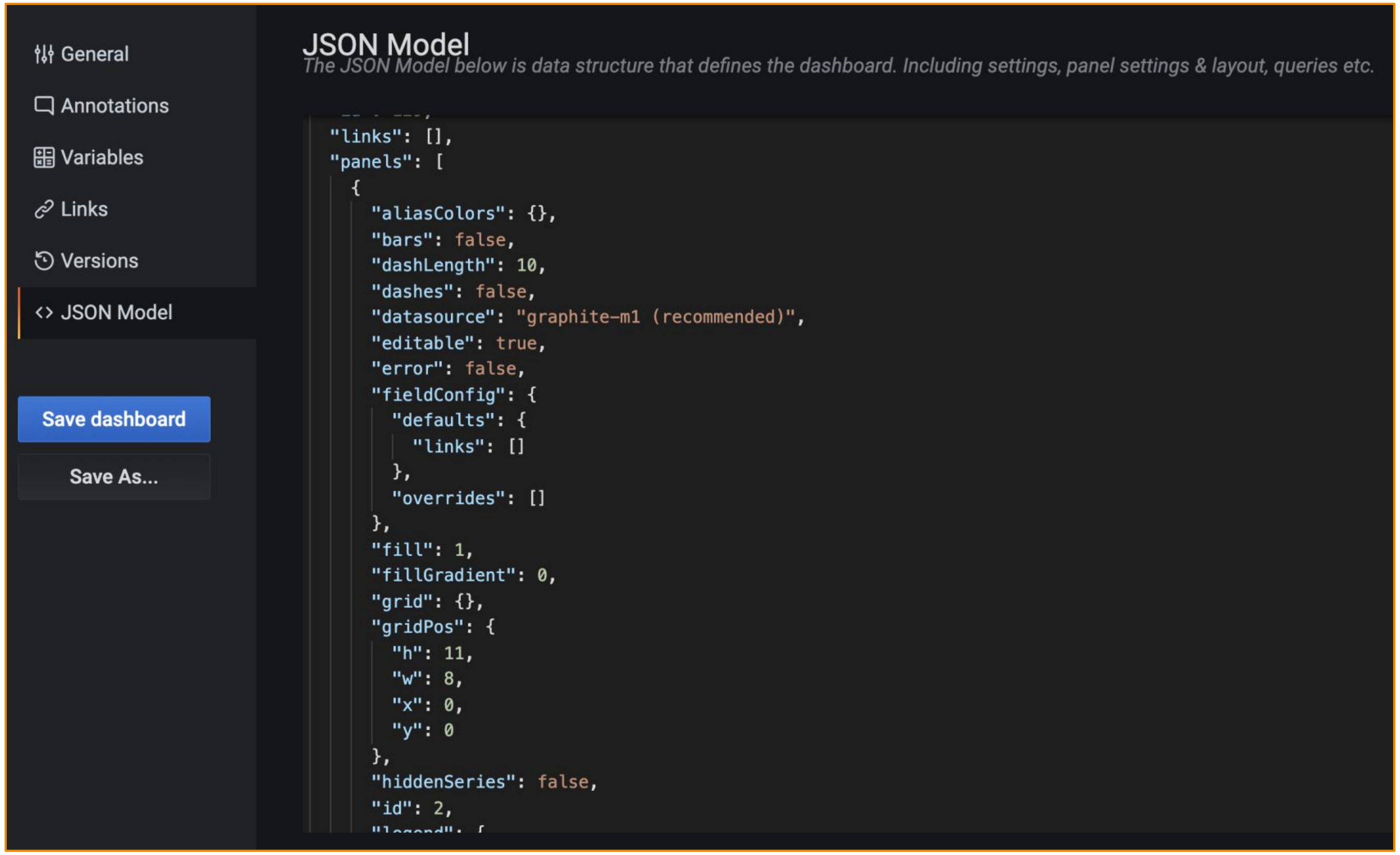
{QSP QiwiServerParty

Dashboard as a code

Антон Юрьев

Grafana JSON Model

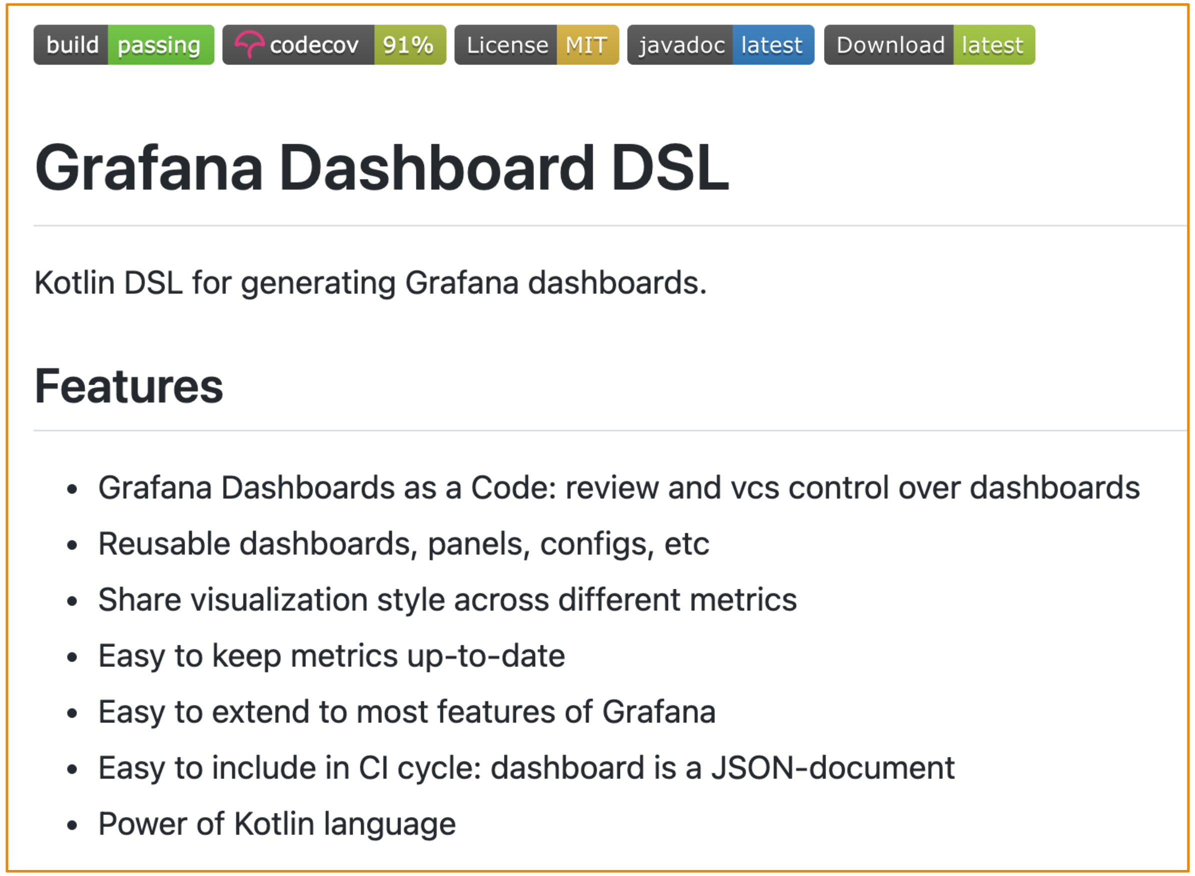


JSON Model

The JSON Model below is data structure that defines the dashboard. Including settings, panel settings & layout, queries etc.

```
"links": [],
"panels": [
  {
    "aliasColors": {},
    "bars": false,
    "dashLength": 10,
    "dashes": false,
    "datasource": "graphite-m1 (recommended)",
    "editable": true,
    "error": false,
    "fieldConfig": {
      "defaults": {
        "links": []
      },
      "overrides": []
    },
    "fill": 1,
    "fillGradient": 0,
    "grid": {},
    "gridPos": {
      "h": 11,
      "w": 8,
      "x": 0,
      "y": 0
    },
    "hiddenSeries": false,
    "id": 2,
```

Grafana Dashboard DSL



Grafana Dashboard DSL

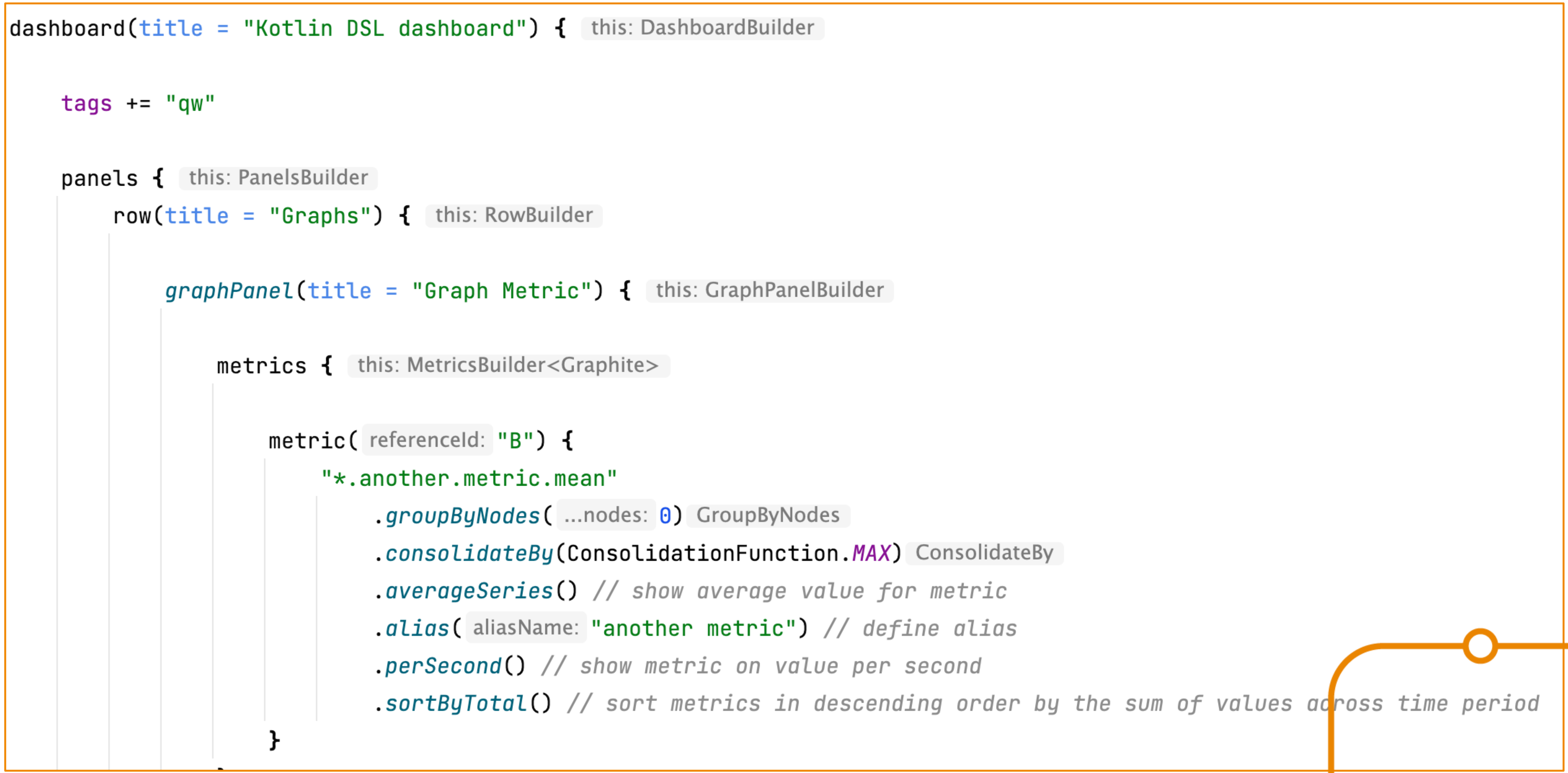
Kotlin DSL for generating Grafana dashboards.

Features

- Grafana Dashboards as a Code: review and vcs control over dashboards
- Reusable dashboards, panels, configs, etc
- Share visualization style across different metrics
- Easy to keep metrics up-to-date
- Easy to extend to most features of Grafana
- Easy to include in CI cycle: dashboard is a JSON-document
- Power of Kotlin language

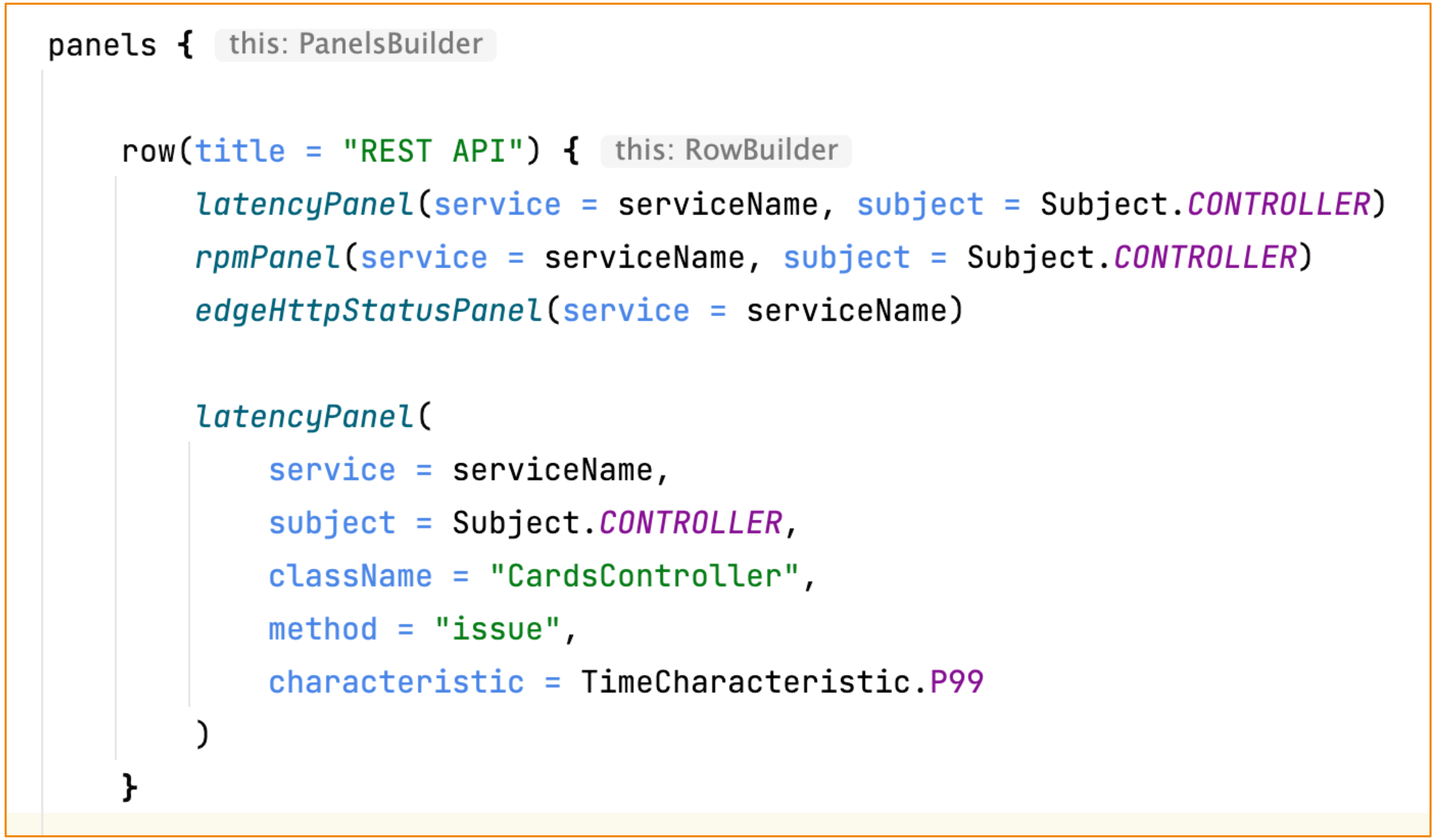
https://github.com/yoomoney/grafana-dashboard-dsl

Kotlin DSL



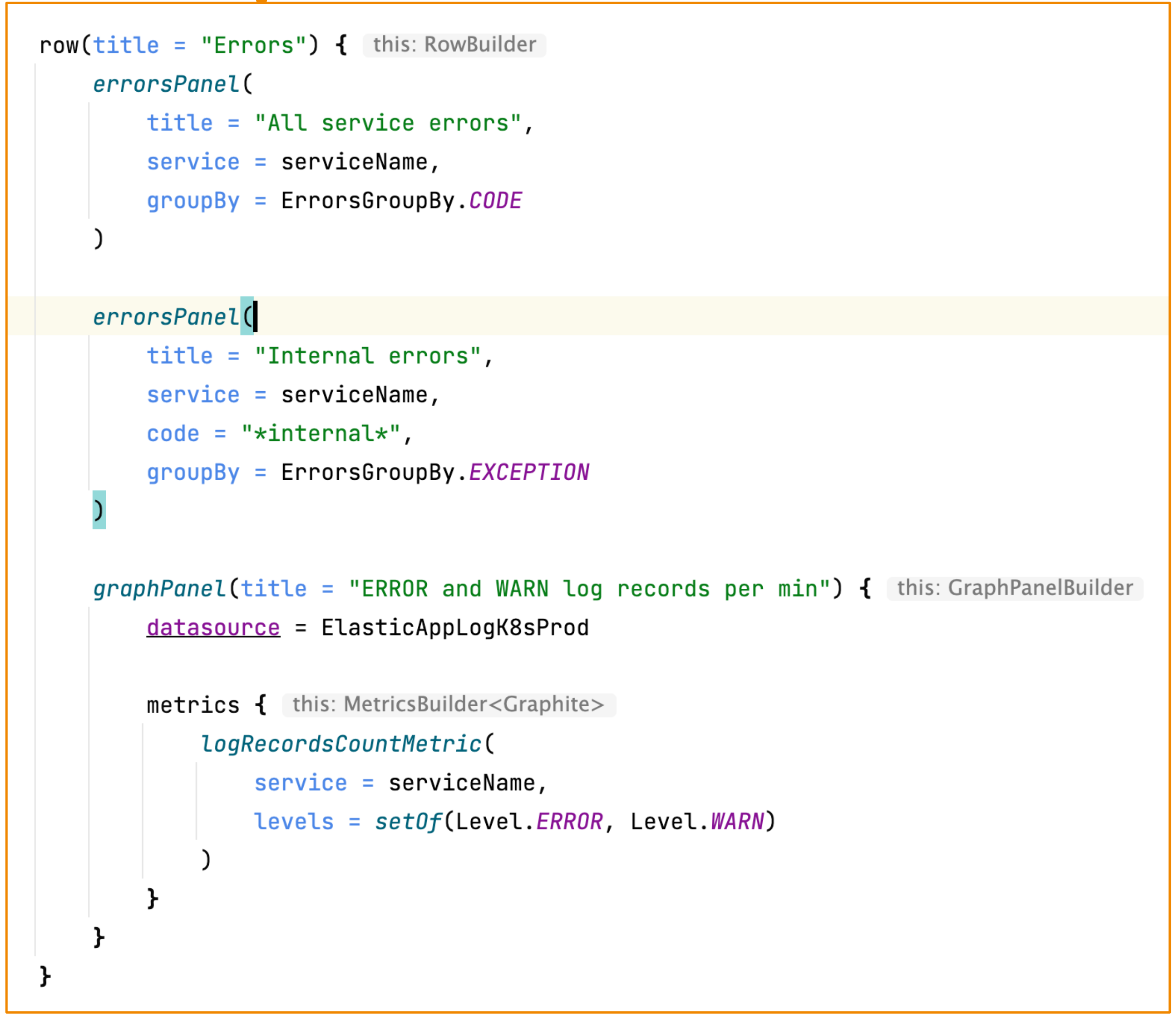
```
dashboard(title = "Kotlin DSL dashboard") {  this: DashboardBuilder

    tags += "qw"

    panels {  this: PanelsBuilder
        row(title = "Graphs") {  this: RowBuilder

            graphPanel(title = "Graph Metric") {  this: GraphPanelBuilder

                metrics {  this: MetricsBuilder<Graphite>

                    metric( referenceId: "B") {
                        "*.another.metric.mean"
                            .groupByNodes( ...nodes: 0) GroupByNodes
                            .consolidateBy(ConsolidationFunction.MAX) ConsolidateBy
                            .averageSeries() // show average value for metric
                            .alias( aliasName: "another metric") // define alias
                            .perSecond() // show metric on value per second
                            .sortByTotal() // sort metrics in descending order by the sum of values across time period
                    }
```

REST API метрики в Kotlin DSL



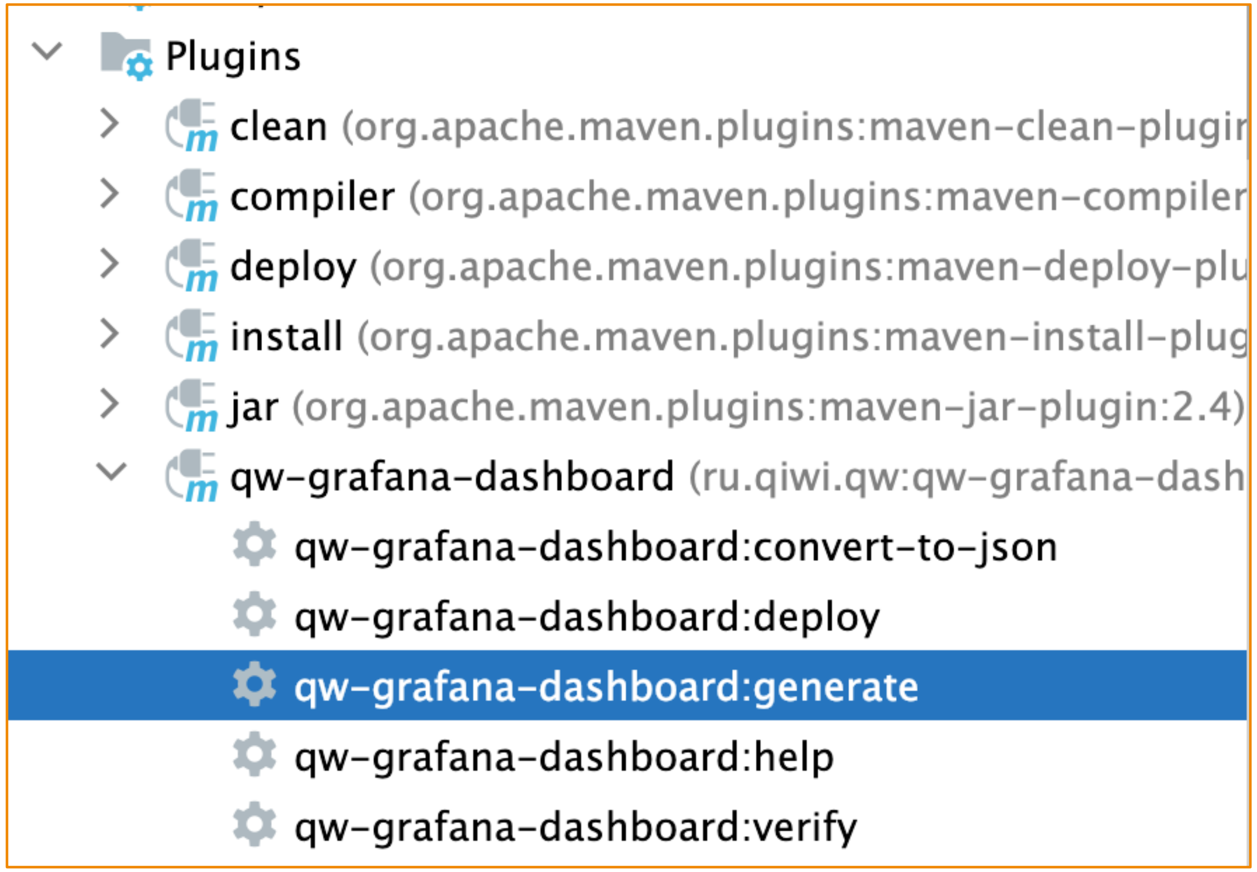
```
panels {  this: PanelsBuilder

    row(title = "REST API") {  this: RowBuilder
        latencyPanel(service = serviceName, subject = Subject.CONTROLLER)
        rpmPanel(service = serviceName, subject = Subject.CONTROLLER)
        edgeHttpStatusPanel(service = serviceName)

        latencyPanel(
            service = serviceName,
            subject = Subject.CONTROLLER,
            className = "CardsController",
            method = "issue",
            characteristic = TimeCharacteristic.P99
        )
    }
```

Метрики ошибок в Kotlin DSL



```
row(title = "Errors") { this: RowBuilder
    errorsPanel(
        title = "All service errors",
        service = serviceName,
        groupBy = ErrorsGroupBy.CODE
    )

    errorsPanel(
        title = "Internal errors",
        service = serviceName,
        code = "*internal*",
        groupBy = ErrorsGroupBy.EXCEPTION
    )

    graphPanel(title = "ERROR and WARN log records per min") { this: GraphPanelBuilder
        datasource = ElasticAppLogK8sProd

        metrics { this: MetricsBuilder<Graphite>
            logRecordsCountMetric(
                service = serviceName,
                levels = setOf(Level.ERROR, Level.WARN)
            )
        }
    }
}
```

Maven plugin



```
Plugins
  clean (org.apache.maven.plugins:maven-clean-plugir
  compiler (org.apache.maven.plugins:maven-compiler
  deploy (org.apache.maven.plugins:maven-deploy-plu
  install (org.apache.maven.plugins:maven-install-plug
  jar (org.apache.maven.plugins:maven-jar-plugin:2.4)
  qw-grafana-dashboard (ru.qiwi.qw:qw-grafana-dash
    qw-grafana-dashboard:convert-to-json
    qw-grafana-dashboard:deploy
    qw-grafana-dashboard:generate
    qw-grafana-dashboard:help
    qw-grafana-dashboard:verify
```

Что стало лучше

- Быстрее настраиваем дашборды и алерты
- Описания дашбордов и алертов хранятся в git и проходят ревью
- Переиспользуем типовые панели и конфигурацию
- Типовой, узнаваемый и понятный всем внешний вид дашбордов
- Больше не забываем настроить алерты для новых микросервисов

Выводы

- Договаривайтесь (convention over configuration)
- Переиспользуйте код в микросервисах
- Автоматизируйте