Code Smells Ana

Métodos diferentes com o mesmo nome  ([GanttDaysOff.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/GanttDaysOff.java))

public boolean isADayOff(GanttCalendar date) {

 return (date.equals(myStart) || date.equals(myFinish) || (date.before(myFinish) && date.after(myStart)));

}

public boolean isADayOff(Date date) {

 return (date.equals(myStart.getTime()) || date.equals(myFinish.getTime()) || (date.before(myFinish.getTime()) && date.after(myStart.getTime())));

}

Métodos que recebem um parâmetro e não o usam ([AlwaysWorkingTimeCalendarImpl.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/AlwaysWorkingTimeCalendarImpl.java))

@Override

public DayType getWeekDayType(int day) {

 // Every day is a working day...

 return GPCalendar.DayType.*WORKING*;

}

@Override

public int getDayMask(Date date) {

 return GPCalendar.DayMask.*WORKING*;

}

@Override

public CalendarEvent getEvent(Date date) {

 return null;

}

Métodos que não fazem nada ([AlwaysWorkingTimeCalendarImpl.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/AlwaysWorkingTimeCalendarImpl.java))

@Override

public void setOnlyShowWeekends(boolean onlyShowWeekends) {

 // Ignore onlyShowWeekends, since weekends are always

 // working days for this calendar

}

@Override

public void setPublicHolidays(Collection<CalendarEvent> holidays) {

}

@Override

public void setBaseCalendarID(String id) {

}

@Override

public void importCalendar(GPCalendar calendar, ImportCalendarOption importOption) {

}

GoF Ana

Facade ([WeekendCalendarImpl.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/WeekendCalendarImpl.java))

private Date getRecurringDate(Date date) {

 myCalendar.setTime(date);

 myCalendar.set(Calendar.*YEAR*,*DUMMY\_YEAR\_FOR\_RECURRING\_EVENTS*);

 return myCalendar.getTime();

}

Factory ([OffsetBuilderImpl.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java))

protected OffsetBuilderImpl(OffsetBuilder.Factory factory) {

 myCalendar = factory.myCalendar;

 myStartDate = factory.myStartDate;

 myViewportStartDate = factory.myViewportStartDate;

 myTopUnit = factory.myTopUnit;

 myBottomUnit = factory.myBottomUnit;

 myDefaultUnitWidth = factory.myAtomicUnitWidth;

 myChartWidth = factory.myEndOffset;

 myWeekendDecreaseFactor = factory.myWeekendDecreaseFactor;

 myEndDate = factory.myEndDate;

 baseUnit = factory.myBaseUnit;

 myRightMarginBottomUnitCount = factory.myRightMarginTimeUnits;

 myOffsetStepFn = factory.myOffsetStepFn;

}

Iterator ([TimelineSceneBuilder.java](https://github.com/anamfrancisco/ganttproject_SE/blob/master/biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/scene/TimelineSceneBuilder.java)) (Método renderTopUnits())

for (Offset nextOffset : topOffsets) {

 if (curX >= 0) {

   TimeUnitText[] texts = myInputApi.getFormatter(nextOffset.getOffsetUnit(), TimeUnitText.Position.*UPPER\_LINE*)

       .format(nextOffset.getOffsetUnit(), curDate);

   final int maxWidth = nextOffset.getOffsetPixels() - curX - 5;

   final TimeUnitText timeUnitText = texts[0];

   textGroup.addText(curX + 5, 0, new TextSelector() {

     @Override

     public Canvas.Label[] getLabels(TextMetrics textLengthCalculator) {

       return timeUnitText.getLabels(maxWidth, textLengthCalculator);

     }

   });

   getTimelineContainer().createLine(curX, topUnitHeight - 10, curX, topUnitHeight);

 }

 curX = nextOffset.getOffsetPixels();

 curDate = nextOffset.getOffsetEnd();

}