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) {

}