# Year in pixels - mood monitoring

Cynthia Alarcón - Patricio Antinao
*Universidad de la Frontera*
c.alarcon16@ufromail.cl - p.antinao02@ufromail.cl

**Resumen**

Se exponen las interfaces gráficas añadidas al proyecto, junto con la descripción de las clases y la relación entre ellas. También se plantean las principales dificultades/lecciones aprendidas, junto a las decisiones tomadas respecto a las cosas que se dijo con anterioridad que se agregarían al proyecto. Se detalla más en profundidad la funcionalidad de cada parámetro/atributo planteado en el diseño de clases.

El proyecto consiste en un programa que permite al usuario registrar su estado de ánimo predominante durante el día, mediante 5 botones de colores (a los cuales denominamos píxeles). La aplicación está orientada principalmente a un público adolescente/juvenil. El programa presenta distintos paneles, cada uno con funcionalidades distintas, por ejemplo uno que despliega un gráfico con la información mensual. Otra pestaña muestra un calendario con los pixeles *coloreados* (registrados) y por colorear. Y además una pestaña de configuración que permite al usuario "personalizar" ciertos aspectos del programa tales como el fondo, el color de los "ánimos" y la hora de notificación.

Keywords: year in pixels, monitoreo de emociones, gráfico de datos, adolescente, aplicación diaria

## 1. Introducción

Es inquietante no saber cómo se está enfrentando emocionalmente al día a día en etapas tan inestables como la adolescencia, es por ello que se ha ideado una forma de que el usuario se detenga unos breves momentos a analizar su día y registrarlo en píxeles para un posterior resumen y conclusión.

La aplicación consiste en:

- Una pestaña de registro: En donde el usuario registrará escogiendo un color representativo de la emoción que siente que ha dominado en su día.

- Una pestaña con un calendario: En el cual aparecerán los días registrados con sus respectivos colores y los que no han sido registrados aún. El formato similar a una matriz, de esta forma el usuario podrá visualizar de forma más agradable sus registros durante el año.
- Una pestaña con opción gráfico: En la cual se despliegan los meses para los cuales es válido realizar una vista en forma de gráfico de los datos que se llevan registrados. De ser así se muestra el gráfico correspondiente al mes seleccionado.

## 2. Justificación

Como fue planteado anteriormente, esta aplicación va a dirigida principalmente a adolescentes, personas que quiza necesiten de un autoconocimiento emocional o bien aquellas que simplemente quieran de alguna manera guardar registro del cómo se van sintiendo en el dia a dia. La idea nació gracias a imágenes publicadas en redes sociales como *instagram*, en las cuales aparecen casillas que representan los días siendo coloreados según el estado de ánimo. El objetivo al que se quiere llegar es básicamente que el usuario tenga un análisis y registro de sus emociones para tener una visión amplia de cómo fue su año, o (más cercano) cómo fué su semana, y con ello lograr que las actividades registradas junto con la emoción sean una guía para cambiar o seguir con los hábitos en la persona *(cabe destacar que lo último mencionado sería una función extra que posiblemente se agregaría más adelante)*.

## 3. Propuesta del diseño de clases

En el modelamiento de la solución se presentan hasta el momento tres paquetes, un paquete que se centra en la solución de la parte lógica “Modelo” del programa, otro paquete en el que se presenta la solución de la parte visual de código y finalmente el paquete en el cual se encuentra la clase principal del programa.

Se deja abierta la posibilidad de agregar un nuevo paquete para futuras entregas (paquete orientado a guardar registro de datos), consideramos que es lo más adecuado para hacer más entendible el modelamiento, ya que de lo contrario el paquete Modelo presentaría muchas clases que están orientadas a lo mismo y sería más conveniente separarlas en un nuevo paquete.

A Continuación se darán a conocer de forma gráfica los paquetes antes mencionados, señalando también cada una de las clases que ellas contienen y resumiendo cuál es la función que éstas tienen dentro del programa.

Comencemos con el paquete Modelo, que como se mencionó anteriormente, es el que da solución a la parte lógica del programa. Veamos su representación.

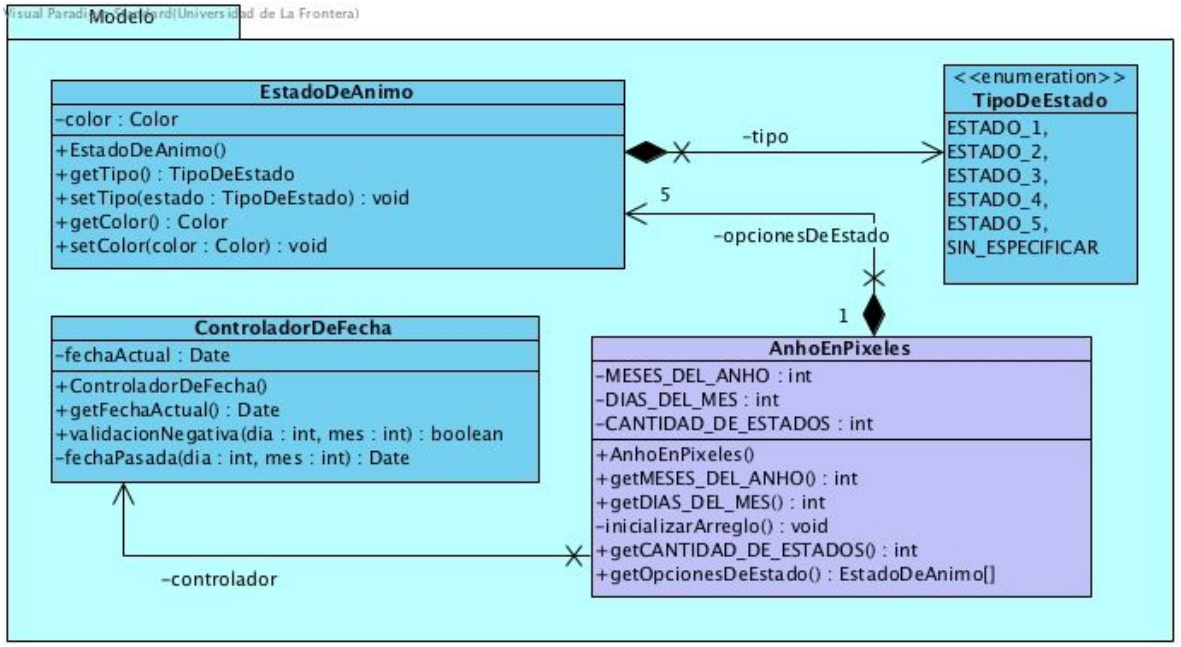


Fig. 1. Diagrama del paquete correspondiente al Modelo.

El paquete Modelo contiene de momento tres clases y una enumeración, siendo la clase AnhoEnPixeles la que presenta mayor relevancia dentro de ésta ya que es en ella donde en cierta forma convergen las demás. Consideramos que Modelo es la base de toda la solución, ya que contiene lo necesario para modelar una posible representación de lo que podríamos llamar un Año en Pixeles. En el se da solución a la validez de las fechas lo cual es muy importante a la hora de guardar registro y se representa el estado de ánimo con sus características visuales.
A Continuación daremos una explicación de cada una de las clases contenidas en éste paquete:

1. *EstadoDeAnimo*: Clase creada de forma estratégica, que modela el estado de ánimo que una persona podría llegar a presentar. En el contexto de la aplicación debemos entender el estado de ánimo como una característica propia de una persona. Esta clase es la representación genérica de un estado de ánimo y presenta los atributos necesarios para su modelamiento. Estados de ánimo hay muchos indudablemente pero para efectos de nuestro programa sólo habrá cinco tipos distintos, que tratan de en cierta forma de englobar al resto. Lo anterior se representa como atributo de la clase y depende de una enumeración que representa los tipos disponibles a elegir.

2. *TipoDeEstado*: Clase enumeración que representa los tipos de estados disponibles en nuestro programa. El ordenamiento de estas es de "mejor a peor" (desde "*ESTADO_1"* hasta "*ESTADO_5"* respectivamente), se incluye además en la enumeración el tipo "SIN_ESPECIFICAR" para aquellos casos que aún no han sido evaluados, por ejemplo cuando el usuario aún no especifica un selección o bien las fechas que aún no transcurren y obviamente no es posible asignar un estado.

3. ControladorDeFecha: Esta es una clase que tiene como principal función actualmente la evaluación de los días, es decir, si sería correcto poder modificar los parámetros en el calendario para un dia cualquiera. Es una clase muy importante ya que nos permite restringir cambios que posiblemente generaría problemas lógicos en la representación.

4. AnhoEnPixeles: Como se mencionó con anterioridad, ésta es posiblemente la clase que más relevancia tiene dentro de este paquete ya que es la que permite la interacción de Modelo con las clases que pertenecen al paquete Visual. Esta clase presenta como atributo los elementos necesarios para el desarrollo lógico del problema. Más adelante se enfatiza el vínculo de ésta clase con el paquete Visual.

Pasemos ahora al paquete Visual, paquete en el cual se aborda la solución visual del programa. El paquete Visual contiene seis clases, cinco de ellas correspondientes a la representación individual de paneles que facilitan la interacción con el usuario y una clase Ventana de tipo JFrame que será la contenedora de la gran mayoría de las demás.
De momento la representación visual se realiza con la visualización de un frame el cual contiene al resto dentro de un conjunto de pestañas que facilita el acceso al resto de forma intuitiva. Pero el representar lo anterior con pestañas podría cambiar en futuras entregas.
Al analizar de forma rápida el paquete podemos observar que Ventana sobresale sobre las demás, principalmente por el vínculo que existe con la demás clases del paquete. Estos vínculos son de tal manera que la ventana no existe sin la presencia de las demás clases (composición).

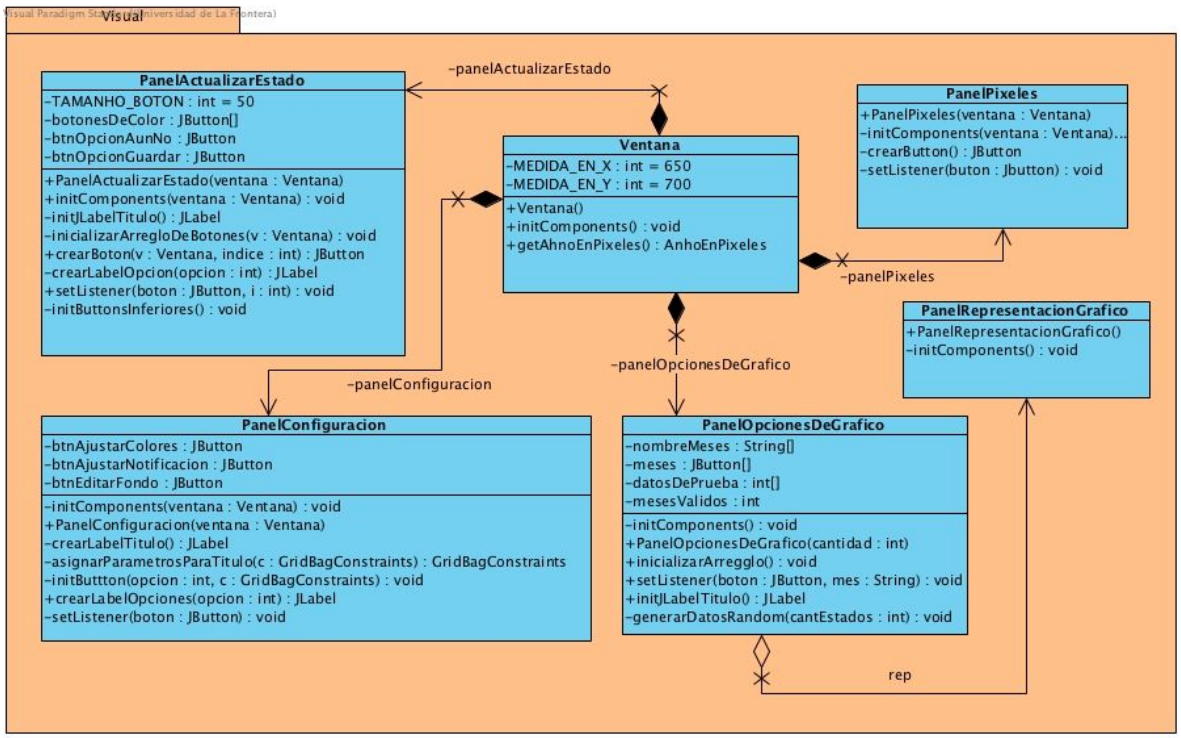

Fig. 2. Diagrama del paquete Visual.

Analicemos de forma individual cada una de las clases:

1. PanelActualizarEstado: Esta clase nos permitirá como dice su nombre actualizar el estado de ánimo para el dia actual, este panel presenta de forma atractiva e intuitiva las opciones, de tal manera que no sea algo complicado de realizar, posterior a la selección de una de las opciones, se realizar el cambio en el calendario que guarda registro de los estados de ánimos y los eventuales registros estadísticos que vendrán en futuras entregas.

2. PanelPixeles: Clase que se encarga de la representación del calendario anual, la representación se asemeja a un matriz, donde cada elemento contenido en ella tendrá un color. Lo anterior siempre que se le haya actualizado un estado de ánimo. Esta clase brinda la posibilidad de actualizar el estado de ánimo para el dia, sin recurrir al PanelActualizarEstado, solo seleccionando el dia actual dentro de la matriz, ya que la acción que desplegará un diálogo del que no se hace mención de momento en el paquete Visual que solo muestra los colores para actualizarlos en el calendario.

3. PanelOpcionesDeGrafico: Este panel muestra los meses del año para los cuales es posible visualizar un gráfico. Los gráficos son de de barras y muestran la cantidad de veces que un estado fue escogido mensualmente.
4. PanelRepresentacionGrafico: Esta clase hace visible un gráfico de barras para lo que se lleva actualizado hasta el momento (grafica mensual), para futuras entregas se implementara un sistema de registro de datos, ya que este panel necesita de información para poder graficar. Se estima que para posteriores entregas hayan cambio respecto a esta clase, ya que mostrará estadísticas más detalladas y la posibilidad de representación de datos anuales, una limitante a esto fue justamente el acceso a registros.

5. PanelConfiguracion: En este panel están contenidos elementos con los cuales se brindan opciones de configuraciones básicas, como por ejemplo la de cambiar los colores de representación para cada estado de ánimo, configurar una posible notificación que cumpla la función de recordar que es necesario actualizar el estado o algo tan basico como editar el fondo de la ventana de actualización. De momento estas opciones son preliminares, podrían haber más en futuras entregas o eventualmente alguna de las anteriores podría no estar presente.

6. Ventana: Clase de mayor relevancia dentro de este paquete, ya que como se mencionó con anterioridad es la contenedora de todas las demás clases del paquete Visual. Esta clase además es la que genera el vínculo con la parte lógica del programa (paquete Modelo), el vínculo e interacción de ambos packages es necesario pues la interacción visual requiere información del Modelo.

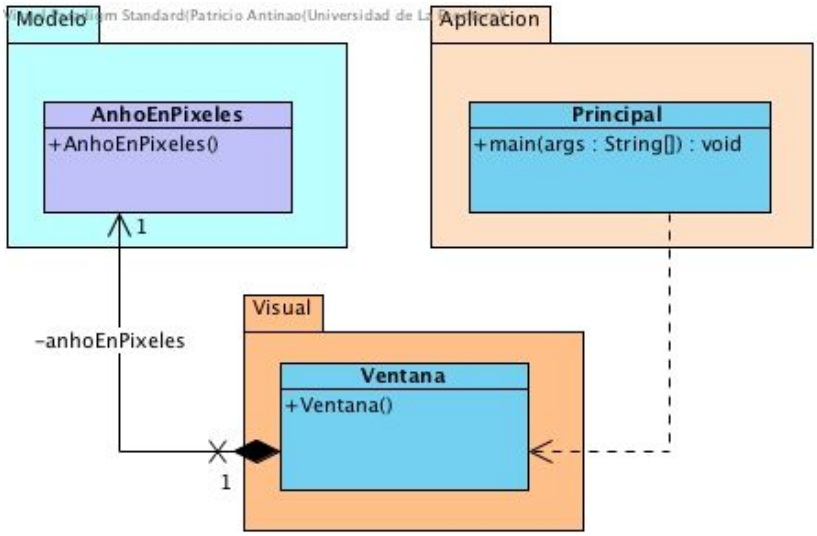


Fig. 3. Diagrama simple de representación de vínculos.

El anterior diagrama es una forma estratégica de representar el vínculo que existe entre los distintos paquetes presentes hasta el momento, ignorando las clases que no presentan relevancias en los vínculos entre paquetes. La interacción de las clases presentes en los paquetes es gracias al vínculo que existe entre ellos.

Para efectos de simplificar la explicación se representó el diagrama general de paquetes de forma simple como en la Fig. 3. y de esta forma solo se evalúan las clases más significativas de cada una de los paquetes y que además son la que generan las interacción entre los paquetes.

Analicemos nuevamente las clases más significativas de los paquetes Modelo y Visual (las responsables de los vínculos):

1. AnhoEnPixeles: Clase que contiene lo necesario "lógicamente hablando" para dar una solución preliminar al problema, en ese sentido si yo generará una instancia de esta clase podría evaluar el tiempo y eventualmente setear un estado, ahora solo necesitamos generar la interacción con el usuario para realizar lo anterior de forma efectiva. Aquí entra la clase Ventana.

2. Ventana: Ventana nos brinda todo lo necesario para que la interacción con el usuario se de efectivamente, es decir el usuario interpreta los textos y opciones presentes en las ventanas y el programa haciendo uso de los atributos de AnhoEnPixeles resuelve el "problema", generando el calendario. Tenemos entonces que, la interacción entre ambas clases es estricta, es decir no puede existir Ventana si no presenta como atributo un AnhoEnPixeles (composición), ya que es de esta última desde donde se hacen las consultas para registrar algún cambio en el calendario anual.

3. Principal: Es la clase principal del programa, aquí se instancia una Ventana y se ejecuta el programa.

## 4. Propuesta del diseño de interfaces gráficas

A continuación se explicará detalladamente el diseño de interfaz gráfica, sujeto a cambios menores.

Al iniciar el programa, lo primero que verá el usuario es el **JPanel** denominado "Día" (ver fig. 4). Aquí es donde el usuario registrará (o no) su estado de ánimo predominante durante el día *(se tiene planeado añadir una contraseña al momento de guardar el día, para evitar así el registro de personas ajenas al monitoreo)*.

Si acciona un color y no presiona "guardar mi día" simplemente se ignorará dicha acción, en cambio, si presiona el botón "guardar mi día" se añadirá el color accionado al **JPanel** "Calendario" (ver fig.5)

Los cambios que se quieren realizar en esta pestaña son:

- Quitar el botón "Aún no", ya que con las modificaciones realizadas el botón en cuestión queda inservible
- Añadir "caritas/imágenes descriptivas" junto a los botones de colores, ya que se estimó que sería un poco engorroso estar leyendo los estados de ánimo que aparecen en la fig.4
- Junto con lo anterior, el texto descriptivo aparecerá solamente cuando el usuario pase el cursor por encima del botón de color; así tendrá información más detallada respecto a qué botón elegir.


Fig. 4. Pestaña "Día" visualizado en conjunto con las demás pestañas disponibles dentro de la aplicación.

Si el usuario elige la pestaña "Calendario", le aparecerá un panel con una "tabla" de **JButtons** deshabilitados para los días pasados, de modo que no pueda cambiar su estado ya registrado y **JLabels** representando los días y meses, formando así un calendario vinculado con la clase "calendar" de java (ver Fig.5). Aquí es donde se podrá visualizar de manera general el "progreso" guardado de forma más colorida y visual.

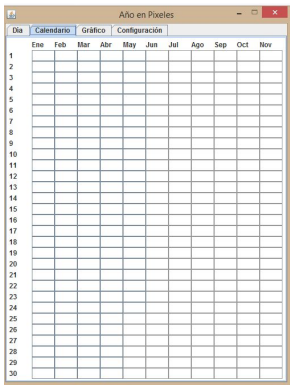


Fig. 5. Pestaña “Calendario” .

Si el usuario elige la pestaña “Gráfico” aparecerá en el **JPanel** botones con la leyenda respectiva de los meses del año (ver fig.6), en donde al presionar uno, por ejemplo “Enero” le aparecerá el gráfico relacionado a los datos guardados dentro del mes de enero (ver fig. 7). Esta ventana le permitirá al usuario ver de forma más detallada los datos guardados.


Fig. 6. Pestaña “Gráfico”

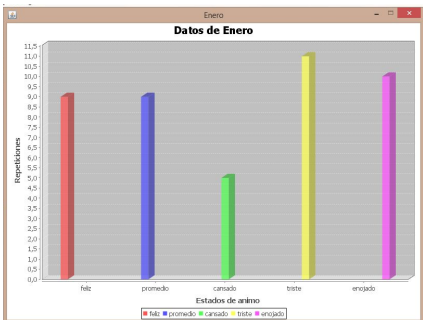

Fig. 7. Ventana adicional que aparece cuando el
usuario presiona el mes el cual quiere consultar.
En ella aparece el gráfico de dicho mes con datos
creados de forma aleatoria para una mejor presentación en el informe

Por último, si el usuario clickea la pestaña "Configuración" aparecerá un panel en donde se le presentarán 3 botones con sus respectivas descripciones, en donde el usuario podrá cambiar los colores de los "píxeles" utilizando la clase JColorChooser, también podrá modificar la hora de notificación/recordatorio para que el usuario ingrese el estado anímico de su día ya acabado, y por último podrá elegir una imagen de fondo para los paneles (ver fig 8), la idea de todo esto es que sea un programa amigable con el usuario *(Aún no se han agregado las funciones de los botones anteriores, por lo que se agregaran a futuro)*

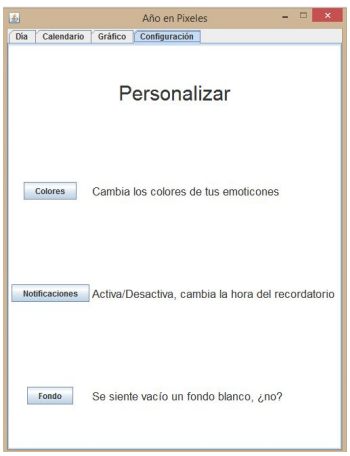

Fig. 8. Pestaña "Configuración"

## 5. Gestión de los datos

Se hace evidente la necesidad de guardar registro de los datos, pensando que el programa no siempre estará en ejecución, en ese sentido guardar los archivos en algún "respaldo" el cual el programa al iniciar la ejecución pueda consultar para restablecer los datos y guardarlos en clases que registren la información. En ese sentido nos falta crear esas clases para las cuales se mencionó en su momento podrían estar ubicadas en un nuevo paquete. Quizá lo ideal sea modelar la semana, el mes y el año con una clase genérica y que sea en ellas donde se guarden los datos una vez inicializada la aplicación y para el término de la misma que los datos se guarden en el archivo sobreescribiendo los datos que allí estaban.

Probablemente sea Json la opción con la cual se dará solución al respaldo de datos. Debido a que el formato es el adecuado y de fácil lectura.

## 6. Principales dificultades/lecciones aprendidas

Las principales dificultades se dieron en la representación visual en general. Ya que cada panel necesita información de otro panel o de alguna clase contenida en el paquete Modelo, entonces el flujo de datos tiene que ser el correcto para que la representación sea coherente. Surge un problema que se hace necesario

solucionar a la brevedad y es la necesidad de guardar registro de los datos que el usuario haya escogido hasta el momento esto con el fin de que al momento de abrir la aplicación estos datos sean solicitados y la representación visual sea en base a esos datos recopilados.

Para el modelamiento de la clase que trabaja con los gráficos se usaron datos aleatorios y la razón es la misma, la falta de datos la cual graficar. Para el modelamiento visual en general son necesarias más clases orientadas al registro de datos para facilitar el flujo de información y no se ensucie el código visual.

## 7. Conclusiones o Comentarios Finales

Aún es necesario agregar mas cosas en futuras entregas, principalmente en el ámbito visual. En el ámbito visual falta mucho por mejorar y es la razón por la cual se quiere implementar la posibilidad de hacer una aplicación personalizable por el usuario. Nos sentimos con la necesidad de hacer algo visualmente atractivo y creemos que es lo que demanda el usuario.

## Bibliografía

-Tooltip:
https://docs.oracle.com/javase/10/docs/api/javafx/scene/control/Tooltip.html

-Date Class
https://docs.oracle.com/javase/10/docs/api/java/sql/Date.html

-Graphics Class
https://docs.oracle.com/javase/10/docs/api/javafx.graphics-summary.html