

# Bhoo!king

Documentación Técnica – Bhoo!king

*Ignacio Fernández Suárez*

# Índice

## Lógica

- Juego ..... páginas 1 al 3
- Reserva ..... páginas 3 al 5

## Interfaz

- Inicio ..... páginas 5 al 7
- Registro ..... páginas 7 al 8
- Pasarela ..... páginas 9 al 10
- Juego ..... páginas 10 al 14
- Canjeo ..... páginas 14 al 15

Wireframe comparativa Original (páginas 15 al 21)

Pruebas de Usuario (páginas 22 al 23)

## ¡Importante! Para el uso correcto de la aplicación

Para visualizar correctamente la aplicación se recomienda cambiar la resolución de Eclipse en Eclipse.exe Propiedades → Compatibilidad → Resolución alta para equipos → Sistema

Además, activar el volumen 🔊, para apreciar la banda sonora del juego.

## Lógica

### Juego

En cuanto a la lógica de la aplicación, la aplicación está organizada mediante los siguientes paquetes con las siguientes clases. A continuación, se explicarán en más detalle las más relevantes.

- ✓ bhooking
  - > Main.java
- ✓ bhooking.auxiliar
  - > Detalles.java
  - > Registro.java
- ✓ bhooking.exceptions
  - > ParseException.java
  - > PaymentException.java
- ✓ bhooking.game
  - > Board.java
  - > Casilla.java
  - > Game.java
- ✓ bhooking.loader
  - > BhookingLoader.java
- ✓ bhooking.model
  - > Castle.java
  - > Descuento.java
  - > Encantamiento.java
  - > User.java
- ✓ bhooking.parser
  - > BhookingParser.java
- ✓ bhooking.service
  - > Bhooking.java
  - > Catalogo.java
  - > Pedido.java

Para las direcciones postales y el sistema de puntuación de calabazas que abarca del 0 al 5 en fracciones de 0.5 se utiliza detalles. Esté, carga y procesa los datos del documento *details.dat*

```
Detalles.java  details.dat x
1 ESC000;Plaza Mayor 78, 33005;3.5
2 ESC001;Calle de las Sombras 13, 28001;3.0
3 ESC002;Rue Mystique 42, 75001;2.5
4 ESC003;Autumn Lane 25, 1010;4.5
5 ESC004;Coastal Retreat 7, 11521;1.5
6 ESC005;Villa Gardens 15, 00185;4.5
7 ESC006;Elegant Manor Street 10, 2750;4.5
8 ESC007;Swampside Haven 666, 061323;2.5
9 ESC008;Luxury Boulevard 123, 8001;5.0
10 ESC009;Modern Mansion Drive 5, 10178;5.0

public Detalles() {
    this.direcciones=new ArrayList<String>();
    this.puntuacion=new ArrayList<String>();
    loadDetails();
}

private void loadDetails() {
    try {
        List<String> lines= new FileUtil().readLines(root);
        for(String line:lines) {
            parseLine(line);
        }
    } catch (FileNotFoundException e) {}
}
```

En cuanto al registro de castillos en la plataforma, se ha empleado una clase de tipo enumeración en la que están registrados los códigos de los castillos. Además, esta incluye un método que devuelve el numero de castillos registrados en la base del programa.

```
import java.util.ArrayList;

public enum Registro {
    ESC000, ESC001, ESC002, ESC003, ESC004, ESC005, ESC006, ESC007,
    ESC008, ESC009;

    public static List<Registro> getPosibilities() {
        return new ArrayList<>(Arrays.asList(values()));
    }
}
```

La lógica del juego tiene un planteamiento que, aunque parezca complejo, es de lo mas sencillo e intuitivo a la hora de programar. Dado que el tablero tiene una forma piramidal un tanto peculiar, se ha optado por crear un tablero de dimensiones (5 filas, 7 columnas) en el cual se han mapeado zonas restringidas. En cuanto a la creación de lo que en si las casillas y el juego es relativamente mas simple que la creación del tablero. A continuación, las casillas:

```
public final static int Ghostbusters=0;

public final static int Ghost_type_1=1;
public final static int Ghost_type_2=2;
public final static int Ghost_type_3=3;
public final static int Ghost_type_4=4;
public final static int Ghost_type_5=5;

public final static int Ghost_king=6;

public final static int null_area=-1;

private boolean isActive;
private int type;

Casilla(int type) {
    setType(type);
    activate();
}

public boolean isActive() {
    return isActive;
}
```

En el tablero, una vez delimitadas las zonas “restringidas”.

```
private boolean isNullArea(int x,int y) {
    //Restricción fila 2
    if(x==2) {if (y==0 || y==6) {return true;}
    //Restricción fila 1
    }if(x==1) {
        if (y==0 || y==1 || y==5 || y==6) {return true;}
    //Restricción fila 0
    }if(x==0) {
        if(!(y==3)) {return true;
    }
}
return false;
}
```

Crearemos un “saco” con los fantasmas a distribuir, que se añadirán al tablero, pero estos se colocaran en orden, pero los fantasmas de un tipo aleatorio dentro de sus posibilidades, ya que tienen que existir 3 fantasmas de cada tipo, siendo 5 los tipos que hay, un solo rey y los cazas fantasmas en la fila inferior.

Creación del saco:

```
private List<Integer> loadStore() {
    List<Integer> almacen = new ArrayList<Integer>();
    for(int i=Casilla.Ghost_type_1;i<=Casilla.Ghost_type_5;i++) {
        for(int j=0; j<3; j++) {
            almacen.add(i);
        }
    }
    Collections.shuffle(almacen);
    return almacen;
}
```

Distribución en el tablero:

```
private void fillNormalGhost() {
    this.tmp_store = loadStore();
    int size=tmp_store.size();
    while(size>0) {
        for(int i=1; i<ROW_SIZE; i++) {
            for(int j=0; j<COL_SIZE; j++) {
                if(!isNullArea(i,j)) {
                    if(!tmp_store.isEmpty()) {
                        board[i][j]=new Casilla(pickOne());
                        size--;
                    }
                }
            }
        }
    }
}

private int pickOne() {
    int val=tmp_store.get(0);
    tmp_store.get(0);
    tmp_store.remove(0);
    return val;
}
```

## Reserva

Dentro del paquete de modelo, tenemos la clase Castillo, Descuento, Encantamiento y Usuario. La clase Descuento tiene métodos estáticos que permiten grabar en cualquier momento un descuento en el fichero *descuentos.dat*, o por ejemplo comprobar si el usuario tiene un descuento.

```
public class Descuento {
    private List<String> DNIs;
    private List<CódigosDescuento> códigos;

    private final static String root="data/descuentos.dat";

    public enum CódigosDescuento{
        EXTRA10, EXTRA25;
    }

    public Descuento() {
        this.DNIs=new ArrayList<String>();
        this.códigos=new ArrayList<CódigosDescuento>();

        loadDiscounts();
    }
```

```

private void loadDiscounts() {
    try {
        List<String> lines= new FileUtil().readLines(root);
        for(String line:lines) {
            parseLine(line);
        }
    } catch (FileNotFoundException e) {}
}

public static void grabDiscounts(String DNI,CodigosDescuento code) {
    try {
        List<String> l = new ArrayList<String>();
        l.add(DNI+";"+code.toString());
        new FileUtil().writeLines(root, l);
    } catch (IOException e) {}
}

public static void removeDiscounts(String DNI) {
    try {
        new FileUtil().removeLines(root, DNI);
    } catch (IOException e) {}
}

public boolean hasDiscount(String dni) {
    for(int i=0; i<DNIs.size(); i++) {
        if(DNIs.get(i).equals(dni)) {return true;}
    }
    return false;
}

```

En la clase servicio tenemos las clases más importante que son las que administran toda la lógica de la aplicación. La clase Bhooking es la principal y tiene e asociado un pedido y un catálogo. El catálogo se podría cargar una sola vez aunque se reinicie para otro usuario ya que no es necesario cargar otra vez el catalogo con el peso computacional que esto tendría a gran escala. Pero es necesario cargar otra vez la clave servicio ya también vuelve al país por defecto.

```

public Bhooking() {
    this.catalogo=new Catalogo();
    this.pedido=new Pedido();
}

public Catalogo showCatalog() {
    return catalogo;
}

public void changeCastlesLanguage(String actualLocale) {
    this.catalogo=new Catalogo(actualLocale);
}

public boolean finalizarPedido() {
    if(pedido.isComplete()) {
        finished();
        try {
            new FileUtil().writeLines("OUTPUT/reservas.dat", pedido.serialize());
            return true;
        } catch (IOException e) {}
    }return false;
}

```

En la clase Bhooking también se comprueban que las fechas sean correctas, los intervalos, habitaciones, ...

En la clase pedido, calcula el total con y sin descuento.

```

public int getAmountWithoutDiscount() {
    return (int) (castillo.getPrecio()*numHabitaciones*daysDifference(fecha_salida, fecha_entrada));
}

public void calculateAmount() {
    double baseAmount=castillo.getPrecio()*numHabitaciones*daysDifference(fecha_salida, fecha_entrada);

    if (!hasDiscount()) {
        this.amount=baseAmount;
    } else {
        CodigosDescuento cd=new Descuento().getDiscount(usuario.getDNI());
        double discountFactor=(cd==CodigosDescuento.EXTRA25) ? 0.75 : 0.9;
        this.amount = baseAmount * discountFactor;
    }
}

```

También, aunque internamente el precio este en una divisa y la transacción se haga en esta misma, es necesario convertir esta al país del usuario en todo momento ya que una vez se haga el pago, los métodos de pago lo cobrarán en la moneda del país.

```

public static String conversion(String locale,double quantity) {
    switch(locale) {
        case "en": return ((int)(quantity*0.87))+" £";
        case "nor": return ((int)(quantity*11.21))+" NOK";
        case "su": return ((int)(quantity*11.14))+" SEK";
        default: return (int)quantity+" €";
    }
}

```

Existe un método que devuelve los días de diferencia.

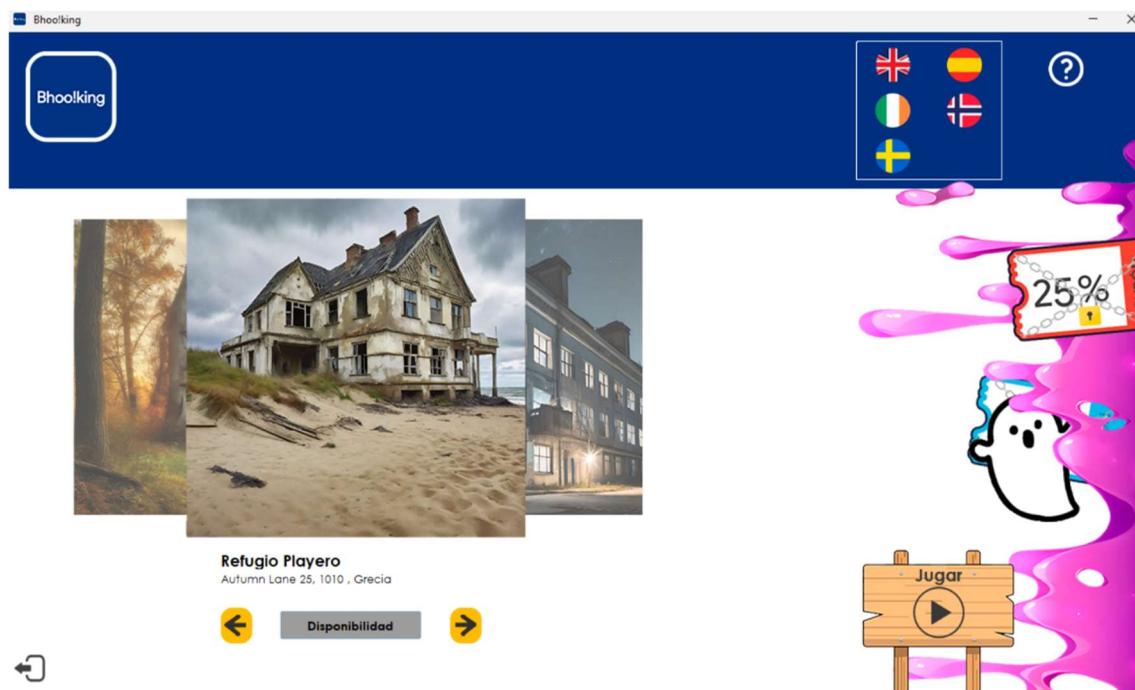
```

protected int daysDifference(Date d1, Date d2) {
    long milis = d1.getTime() - d2.getTime();
    long diasDiferencia = TimeUnit.DAYS.convert(milis, TimeUnit.MILLISECONDS);
    return (int)diasDiferencia;
}

```

## Interfaz

### Inicio



Para la interfaz del inicio, dado que su despliegue se trata en TPV's de una agencia de viajes, mostraríamos los hoteles en un carrusel de imágenes. Este carrusel no existe en la galería de elementos de Java Swing, se ha tenido que crear. Y este será muy importante para la aplicación, siendo el modo de selección del castillo, es decir el índice de este será el que se utiliza para el despliegue de datos.

El carrusel se comporta como un array, pero estableciendo reglas y comportamientos en los límites de este:

```
private void setCarrusel(int pos) {
    int size=carrusuel.length-1;
    indexCarrusel=(pos>=size)? size-1 : (pos<0)? 0 : pos;
    setImagesCarrusel(indexCarrusel);
    setInfoCarrusel(carrusuel[indexCarrusel]);
}

private void setImagesCarrusel(int pos) {
    int size=carrusuel.length - 1;
    int indexIzq=(pos-1<0)? -1 : (pos-1>=size)? size-2 : pos-1;
    int indexCentral=(pos<0)? 0 : (pos>=size)? size-1 : pos;
    int indexDer=(pos+1<0)? 0 : (pos+1>=size)? -1 : pos+1;

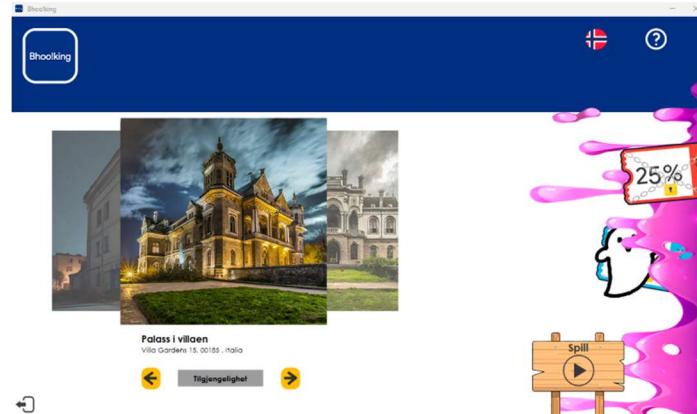
    if (indexIzq===-1) {
        getCarrousel_izq().setIcon(null);
    } else {
        getCarrousel_izq().setIcon(new ImageIcon(getClass().getResource("/img/ESC00")));
    }
    getCarrousel_central().setIcon(new ImageIcon(getClass().getResource("/img/ESC00")));

    if (indexDer===-1) {
        getCarrousel_der().setIcon(null);
    } else {
        getCarrousel_der().setIcon(new ImageIcon(getClass().getResource("/img/ESC00")));
    }
}
```

Se inicia el carrusel aproximadamente por la mitad.

```
private void inicializarCarrousel() {
    List<Castle> catalogo = bhooking.showCatalog().getCatalogo();
    this.carrusuel=catalogo.toArray(new Castle[catalogo.size()]);
    this.indexCarrusel = carrusuel.length/2;
    setCarrusel(indexCarrusel);
}
```

Para el desplazamiento en el carrusel sirven tanto las flechas como tocando las imágenes laterales y para el modo de selección tanto en el botón de disponibilidad como clicando la imagen central.



Para ese efecto de difuminado en las imágenes laterales se han creado dos overlays del mismo tamaño que las imágenes. Estas son simplemente un color blanco con cierta transparencia y además son estos los que reciben las acciones al ser pulsados.

↳ IbOverlayDer - ""

↳ IbOverlayIzq - ""

En la ventana principal también contamos un botón para cerrar la sesión y prepararla para el próximo usuario. También se puede cambiar el país en el panel superior además de desplegar la ayuda.

## Reserva

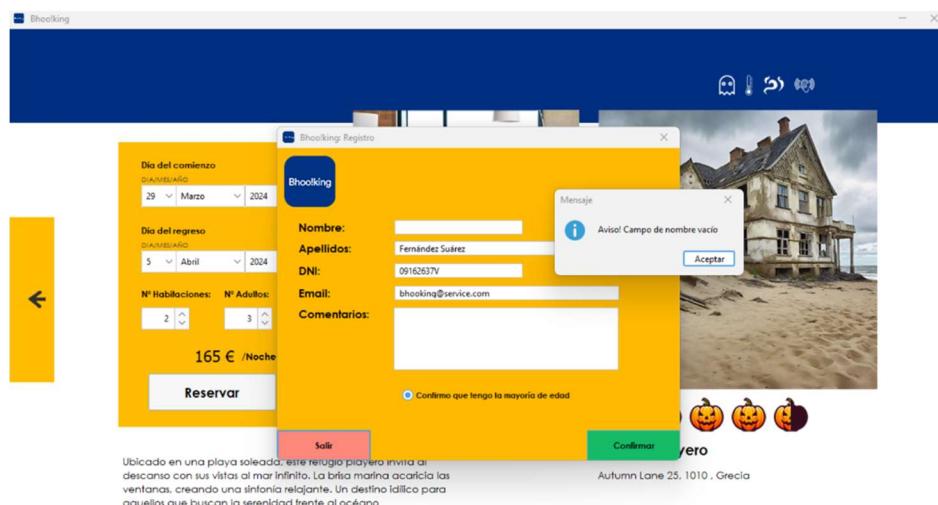
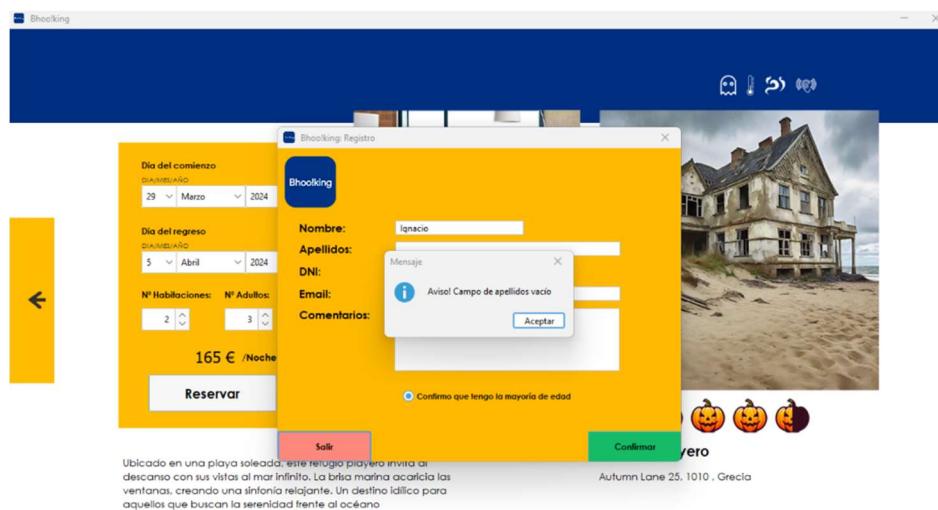
The screenshot shows a search result for a luxury villa. On the left, there is a yellow sidebar with input fields for 'Innsjekkingsdato' (Check-in date) set to '1 Januar 2024', 'Utbjekkingsdato' (Check-out date) set to '1 Januar 2024', 'Antall rom:' (Number of rooms) set to '1', 'Antall voksne:' (Number of adults) set to '1', and a total price of '2522 NOK /Natt'. Below these fields is a 'Reserver' (Book) button. To the right of the sidebar are three images: an interior view of a grand hall, an exterior view of a garden with autumn foliage, and a night view of a large, illuminated villa. Below the images is a row of five small jack-o'-lantern icons. At the bottom, the text reads 'Palass i villaen' and 'Villa Gardens 15, 00185 , Italia'.

The screenshot shows a search result for a beachside refuge. On the left, there is a yellow sidebar with input fields for 'Día del comienzo' (Start date) set to '31 Marzo 2024', 'Día del fin' (End date) set to '5 Abril 2024', 'Nº Habitaciones' (Number of rooms) set to '1', 'Habitación' (Room type) set to 'Bullos: Noche', and a 'Reservar' (Book) button. To the right of the sidebar are three images: an interior view of a bright living room overlooking a beach, an exterior view of a sandy beach with greenery, and an exterior view of a dilapidated, multi-story house on a beach. Below the images is a row of five small jack-o'-lantern icons. At the bottom, the text reads 'Refugio Playero' and 'Autumn Lane 25, 1010 , Grecia'.

En la pagina de reserva se muestran tres imágenes de los palacios, la valoración en calabazas, una descripción, los **encantamientos** en la parte superior derecha, para la cual se ha creado un panel con un FlowLayout, al que se le van añadiendo las imágenes de los encantamientos.

- Esta es la única ventana *resizable* aunque dentro de unos límites, tanto superiores como inferiores. Dado el diseño minimalista y diseño estético de la ventana se ha utilizado como layout del panel base Absolute Layout, por lo que para el completo redimensionamiento de las imágenes debería también cambiar el tamaño de las JLabel. Aun así se ha construido para que las imágenes se adapten dinámicamente al resacarla la ventana al tamaño de la JLabels.
- Esta ventana de la reserva junto a la inicial comparte un Card Layout anfitrión que hace mostrar una u otra. Además, al volver con el botón amarillo lateral de retorno también se restablece el valor de los spinner y resto de componentes. Para mostrar los datos del castillo se emplea el índice del carrusel.

## Registro



En la ventana modal de registro, se asocia un usuario al pedido si este cumple con los requisitos técnicos necesarios. Y si este se completa satisfactoriamente muestra la pasarela de pago.

## Pasarela

**Refugio Playero**

Autumn Lane 25, 1010, Grecia

- Fecha de inicio: 29 marzo 2024
- Fecha de regreso: 05 abril 2024
- Nº Adultos: 3
- Nº Habitaciones: 2

Volver al configurador

**Total a pagar:** 1980 €

Pago seguro garantizado: VISA, Mastercard, PayPal

**Residencia Elegante**

Elegant Manor Street 10, 2750, Portugal

- Fecha de inicio: 01 mayo 2024
- Fecha de regreso: 05 mayo 2024
- Nº Adultos: 1
- Nº Habitaciones: 1

Volver al configurador

**Total a pagar:** 990 €

Pago seguro garantizado: VISA, Mastercard, PayPal

Si el usuario tuviese un descuento, lo descuenta del total y avisa al usuario.

En todo momento se puede volver al configurador y no volvería a pedir al usuario a registrarse. El configurador conservaría los datos actuales.

Un método que cabe destacar es un método que sirve para aliviar la memoria a corto plazo del usuario y es retener las fechas de los días, o en todo caso más

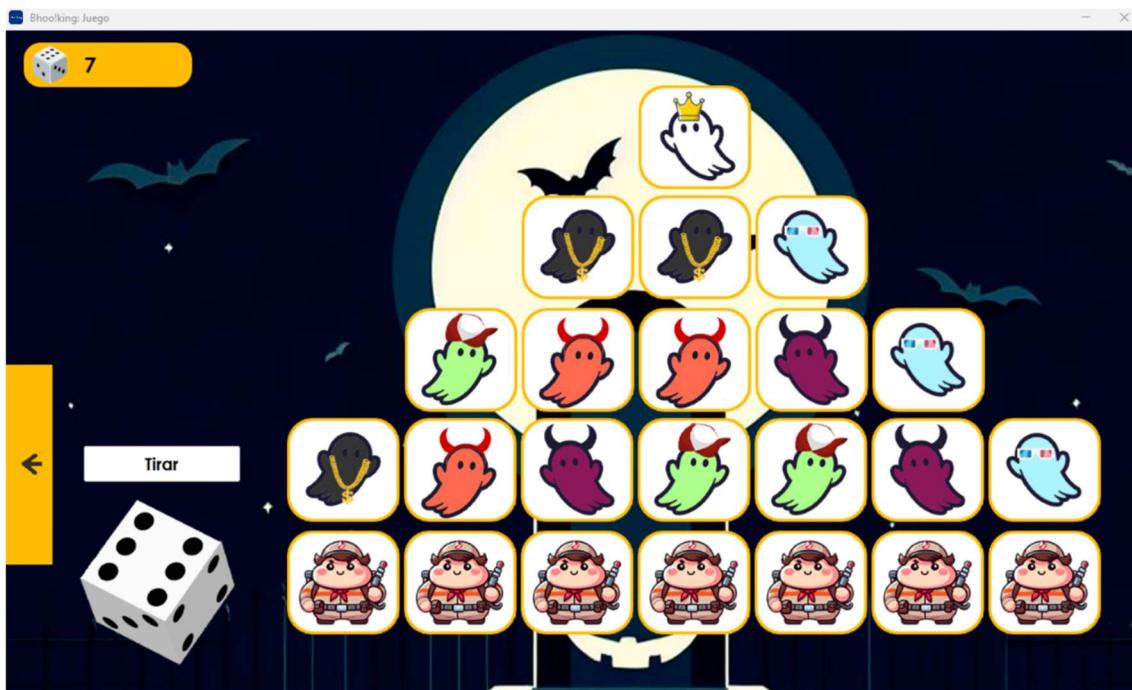
próximas en el combobox. Si el usuario selecciona el 30 de mayo, pero se refiera al 30 de junio, no se vuelve al día 1, se mantiene el 30, pero si ahora se pasa a febrero, pues queda el 29 que es el más cercano.

```
private void updateDays(JComboBox<String> monthComboBox, JComboBox<Integer> yearComboBox,
JComboBox<Integer> dayComboBox) {
int selectedMonth = monthComboBox.getSelectedIndex() + 1;
int selectedYear = (int) yearComboBox.getSelectedItem();
int daysInMonth = getDaysInMonth(selectedMonth, selectedYear);

int saved= (int)dayComboBox.getSelectedItem();
saved=getSavedDay(saved, daysInMonth);
dayComboBox.removeAllItems();
for (int i=1;i<=daysInMonth; i++) {
    dayComboBox.addItem(i);
}
dayComboBox.setSelectedItem(saved);
}

private int getSavedDay(int day,int dayInMonth) {
if(day<=dayInMonth) {
    return day;
} else if(dayInMonth==29) {
    return 29;
}return 28;
}
```

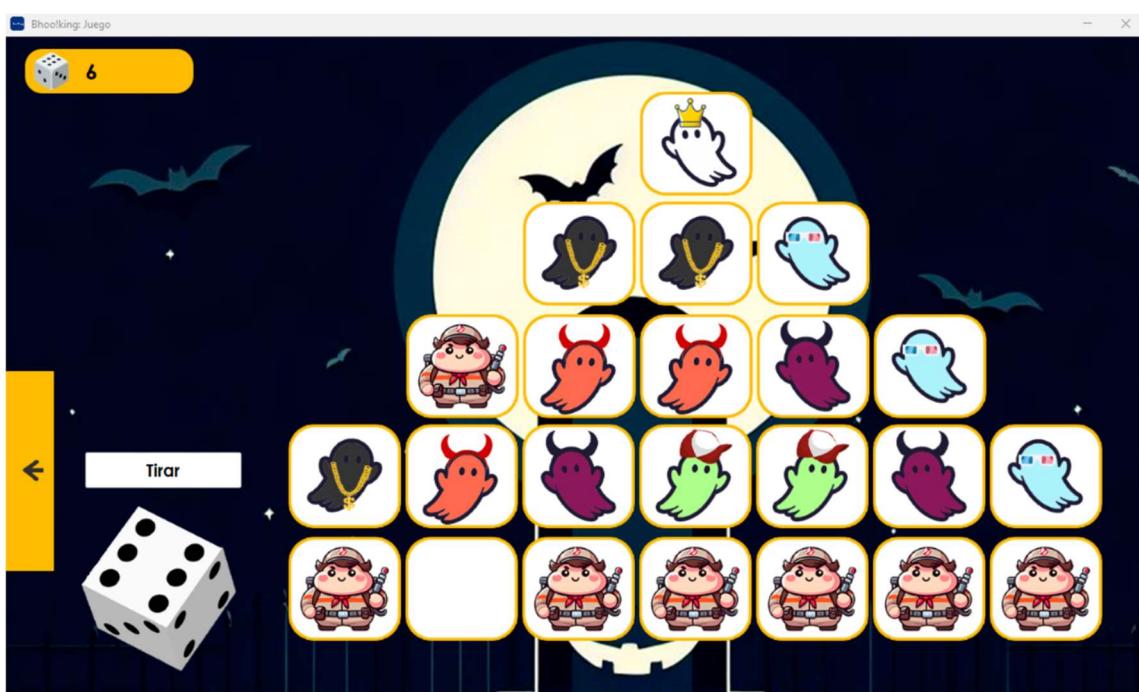
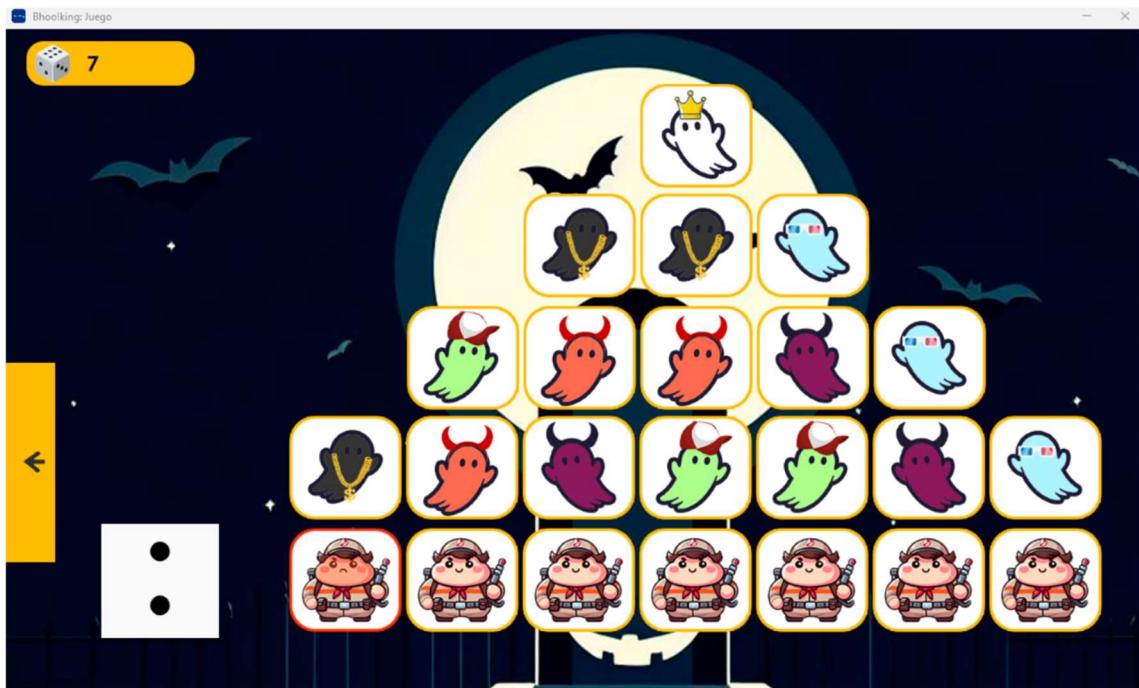
## Juego



Para el juego:

Se puede tocar el botón de Lanzar o tocar la imagen del dado.

- Despues será necesario tocar un cazafantasmas y despues la posición a avanzar.



Se muestra al pasar el ratón con un borde azul, la casilla que se puede mover. Así indica fácilmente al usuario que está en lo correcto.

Se han creado dos clases para procesar las acciones, una para los caza fantasmas y otra para el resto de fantasmas. Con diferentes funcionalidades.

## La de los fantasmas:

```
class ProcesaGhosts extends MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        if(isMovementActive) {
            if (destinoLabel!=null) {
                Coordinate c = new Coordinate(destinoLabel.getName());
                int type= game.getCopyOfBoard()[c.getX()][c.getY()].getType();
                destinoLabel.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_0")));
            }

            JLabel label = (JLabel)e.getSource();

            if (isValidSelection(fuente, new Coordinate(label.getName()))) {
                destinoLabel = label;
                selectDestino(label);
                Coordinate c = new Coordinate(label.getName());
                int type= game.getCopyOfBoard()[c.getX()][c.getY()].getType();
                label.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_0")));
                realizarMovimiento();
            }
        }
    }

    @Override
    public void mouseEntered(MouseEvent e) {
        if (isMovementActive) {
            JLabel label = (JLabel) e.getSource();
            if (isValidSelection(fuente, new Coordinate(label.getName()))) {
                Coordinate c = new Coordinate(label.getName());
                int type= game.getCopyOfBoard()[c.getX()][c.getY()].getType();
                label.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_0")));
            }
        }
    }

    @Override
    public void mouseExited(MouseEvent e) {
        if (isMovementActive) {
            JLabel label = (JLabel) e.getSource();
            Coordinate c = new Coordinate(label.getName());
            int type= game.getCopyOfBoard()[c.getX()][c.getY()].getType();
            label.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_" + type)));
        }
    }
}
```

## La de los cazafantasmas:

```
class ProcesaGhostbusters extends MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        if(isMovementActive) {
            if (fuenteLabel!=null) {
                Coordinate c = new Coordinate(fuenteLabel.getName());
                if(game.getCopyOfBoard()[c.getX()][c.getY()]==null) {
                    fuenteLabel.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_0")));
                }else {
                    fuenteLabel.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_0")));
                }
            }

            JLabel label = (JLabel)e.getSource();
            fuenteLabel = label;
            selectFuente(label);
            fuenteLabel.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_0")));
        }
    }
}
```

Aunque el tablero tenga esa peculiaridad, se han creado todos los botones de forma dinámica, distribuyéndolos por filas y mapeándolos.

- Como los JLabel no tienen setActionCommand, se ha utilizado los métodos setName y getName para mapear. Y una clase interna Coordinate que lo procesa

```
protected class Coordinate {  
    private int x;  
    private int y;  
  
    Coordinate(String coordenada) {  
        String s = coordenada.substring(1, coordenada.length() - 1);  
        String[] parts = s.split(",");  
        x = Integer.parseInt(parts[0]);  
        y = Integer.parseInt(parts[1]);  
    }  
  
    protected int getX() {  
        return x;  
    }  
  
    protected int getY() {  
        return y;  
    }  
}
```

Y el distribuidor por filas:

```
private void crearTablero() {  
    int count=0;  
    for (int i=0;i<Board.ROW_SIZE; i++) {  
        for(int j=0;j<Board.COL_SIZE; j++) {  
            if(game.getCopyOfBoard()[i][j]!=null) {  
                Coordinate c = new Coordinate(getMargenes()[count]);  
                distribuidorFilas(c.getX(), c.getY());  
                count++;  
            }  
        }  
    }  
}  
  
private void distribuidorFilas(int row,int col) {  
    if(row==0) {  
        getPnFila1().add(crearEtiqueta(row,col));  
    }if(row==1) {  
        getPnFila2().add(crearEtiqueta(row,col));  
    }if(row==2) {  
        getPnFila3().add(crearEtiqueta(row,col));  
    }if(row==3) {  
        getPnFila4().add(crearEtiqueta(row,col));  
    }if (row==4) {getPnFila5().add(crearEtiqueta(row,col));  
    }validate();  
}
```

Se mapean asignando por ejemplo a cada posición del elemento y dada su fila, su posición general respecto al tablero con una posición en el mismo.

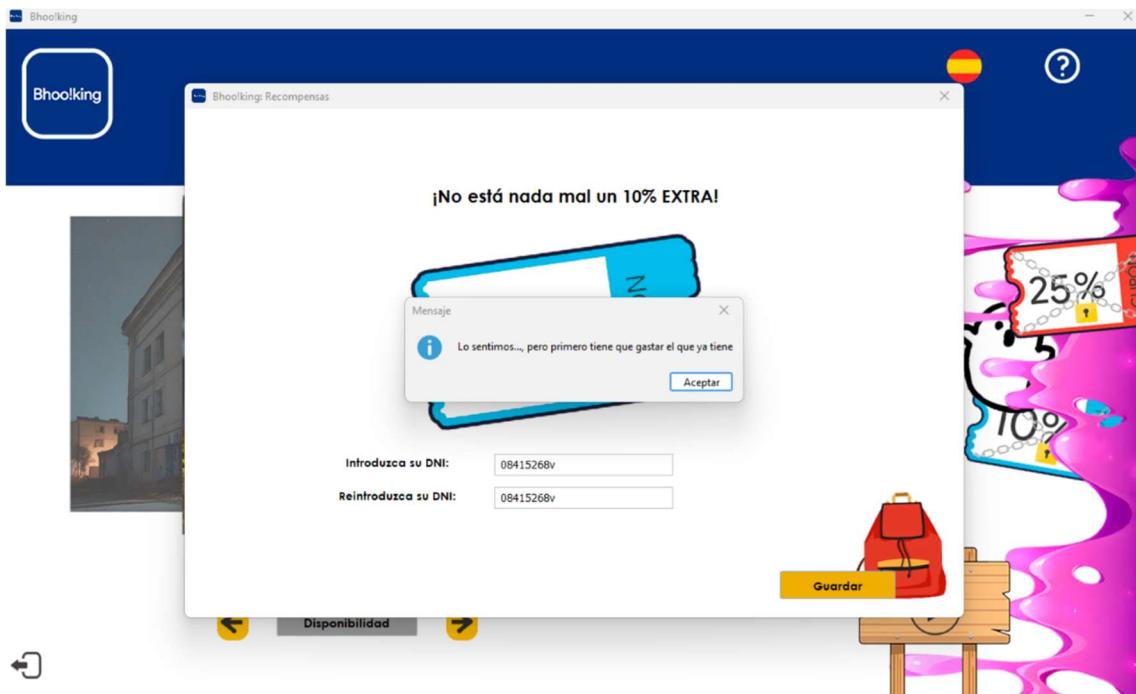
```

private JLabel crearEtiqueta(int row, int col) {
    JLabel label = new JLabel("");
    label.setName("(" + row + "," + col + ")");
    int type = game.getType(row, col);
    if (type!=Casilla.null_area) {
        label.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_"+type+".png")));
        if(type==Casilla.Ghostbusters) {
            label.addMouseListener(pgb);
        }else {
            label.addMouseListener(pg);
        }
    } else {
        label.setIcon(new ImageIcon(VentanaGame.class.getResource("/img/Casilla_null.png")));
    }
    return label;
}

private String[] getMargenes() {
    return new String[] { "(0,3)", "(1,2)", "(1,3)", "(1,4)", "(2,1)", "(2,2)", "(2,3)", "(2,4)", "(3,0)", "(3,1)", "(3,2)", "(3,3)", "(3,4)", "(3,5)", "(3,6)", "(4,0)", "(4,1)", "(4,2)", "(4,3)", "(4,4)", "(4,5)", "(4,6)"};
}

```

## Canjeo



Una vez finalizado el juego, en la pestaña de canjeo del cupón, habrá 3 opciones. No haber ganado ningún premio, el cupón EXTRA10 y EXTRA25. Además, si el usuario no lo ha gastado todavía no podrá almacenarlo. Sin embargo una vez este sea gastado si podrá guardarse en el `descuentos.dat`.

Bhoolking

Bhoolking: Recompensas

Lo sentimos..., pero no ha obtenido ningún cupón, ¡sigue intentando!

Cerrar

Disponibilidad

?

Bhoolking

Bhoolking: Recompensas

¡No está nada mal un 10% EXTRA!

10%

CUPÓN

Introduzca su DNI: 08415268v

Reintroduzca su DNI: 08415268v

Guardar

?

Bhoolking

Bhoolking: Recompensas

¡Fantástico un 25% EXTRA!

25%

CUPÓN

Introduzca su DNI:

Reintroduzca su DNI:

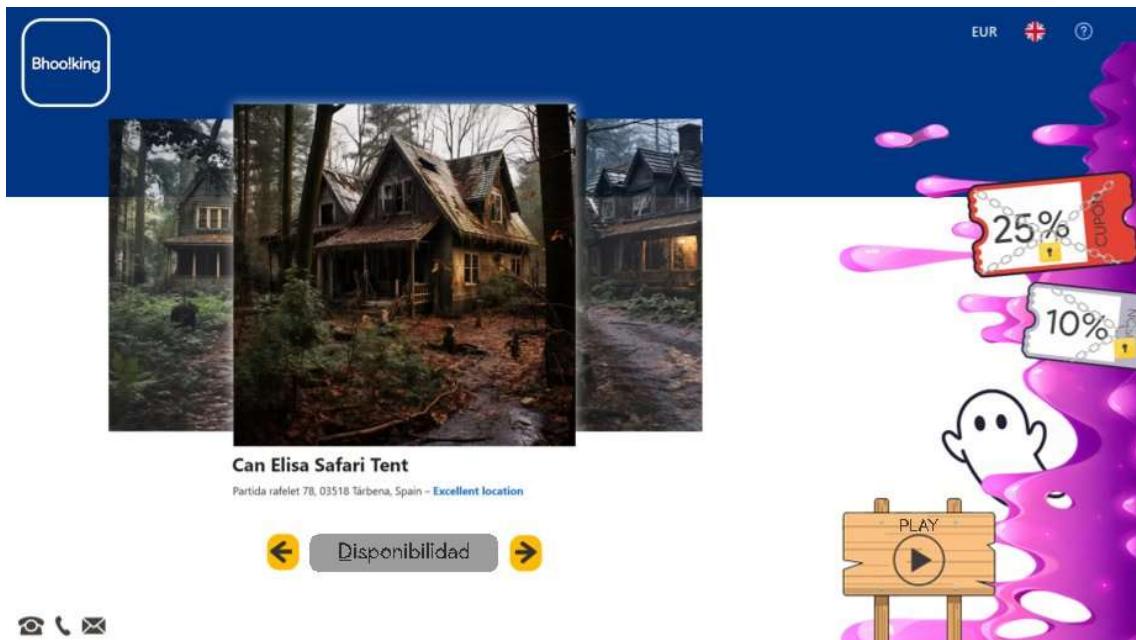
Guardar

?

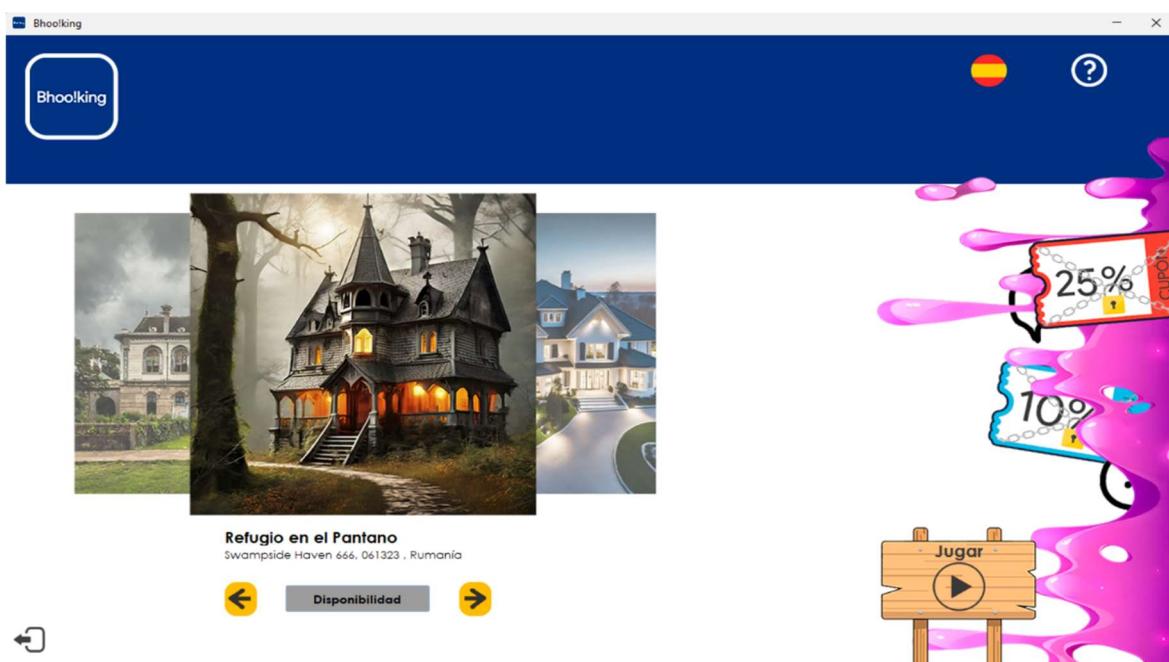
## Wireframe Original Comparativa

Finalmente, la aplicación se parece muchísimo a al Wireframe original, pero con varias mejoras sustanciales.

### WIREFRAME



### REAL



Respecto a la página original, la distribución y estilos de los botones es prácticamente igual. Cambios insignificantes que pueden notarse son, el color del tamaño del cupón de 10 y el fantasma que se ha sustituido por un gif.

## WIREFRAME

The wireframe shows a dark blue header bar. Below it, on the left, is a sidebar with a yellow background containing form fields for 'Check-in date' and 'Check-out date', both with dropdown menus. Below these are dropdowns for '2 adults - 0 children - 1 room'. At the bottom of the sidebar is a white button labeled 'RESERVAR' with the price '230 €/noche' underneath. On the right side of the wireframe are two images: one of an interior room with large windows and another of a rustic wooden cabin in a forest at night.

**Lore ipsum** es simplemente el texto de relleno de las imprentas y archivos de texto. Lore ipsum ha sido el texto de relleno estándar de las industrias desde el año 1500, cuando un impresor (N. del T. persona que se dedica a la imprenta) desconocido usó una galería de textos y los mezcló de tal manera que logró hacer un libro de textos especímenes. No sólo sobrevivió 500 años, sino que también ingresó como texto de relleno en documentos electrónicos, quedando esencialmente igual al original. Fue popularizado en los 60s con la creación de las hojas "Letraset", las cuales contenían pasajes de Lore ipsum, y más recientemente con software de autoedición, como por ejemplo Aldus PageMaker, el cual incluye versiones de Lore ipsum.

### Can Elisa Safari Tent

Partida rafelet 78, 03518 Tárrega, Spain – [Excellent location](#) - [show map](#)



## REAL

The screenshot shows a window titled 'Bhoolking'. The sidebar on the left contains form fields for 'Día del comienzo' (Check-in date) and 'Día del regreso' (Check-out date), both with dropdown menus. Below these are dropdowns for 'Nº Habitaciones' (Number of rooms) and 'Nº Adultos' (Number of adults). At the bottom of the sidebar is a white button labeled 'Reservar' with the price '165 € /Noche' underneath. To the right of the sidebar are three images: one of an interior room overlooking a beach, one of a sandy beach, and one of a large, dilapidated building on a beach. Below each image is a row of five small jack-o'-lantern icons.

Ubicado en una playa soleada, este refugio playero invita al descanso con sus vistas al mar infinito. La brisa marina acaricia las ventanas, creando una sinfonía relajante. Un destino idílico para aquellos que buscan la serenidad frente al océano.

### Refugio Playero

Autumn Lane 25, 1010 , Grecia

La pantalla de reserva es casi idéntica, se ha cambiado la disposición de los encantamientos y no se ha implementado dada a su complejidad el mapa.

## WIREFRAME

The wireframe shows a yellow registration form overlaid on a blurred background image of a beach house. The form includes fields for DNI, Password, Nombre, Apellidos, DNI, Email de contacto, Password, and Confirmación Password. It also features a '¿Está ya registrado?' checkbox and 'Salir' and 'Confirmar' buttons.

Check-in date:  Check-out date:   
2 adults - 0 children  
RESERVA 230 €  
Lorem Ipsum es simplemente un texto falso que se ha usado como material de impresión y archivos de texto estandarizado de los últimos siglos. Es el tipo de persona que se usa para probar una galería de textos y páginas web. Es igual a un libro de textos en que también incluye imágenes electrónicas, quedando popularizado en los últimos años en páginas con software de autoedición, como por ejemplo el software que incluye versiones de Lorem Ipsum.

Check-in date:  Check-out date:   
2 adults - 0 children  
RESERVA 230 €  
Lorem Ipsum es simplemente un texto falso que se ha usado como material de impresión y archivos de texto estandarizado de los últimos siglos. Es el tipo de persona que se usa para probar una galería de textos y páginas web. Es igual a un libro de textos en que también incluye imágenes electrónicas, quedando popularizado en los últimos años en páginas con software de autoedición, como por ejemplo el software que incluye versiones de Lorem Ipsum.

## REAL

The real screenshot shows the same registration form as the wireframe, but with a blue header and a sidebar on the left containing travel details. The sidebar includes fields for check-in and check-out dates, number of guests, and a price of 165 €/night. A 'Reservar' button is at the bottom. The main form has fields for Nombre, Apellidos, DNI, Email, and Comentarios. It includes a checkbox for age confirmation and 'Salir' and 'Confirmar' buttons. The background image is a beach house.

Día del comienzo:  DIA/MES/AÑO:  29 Marzo 2024  
Día del regreso:  DIA/MES/AÑO:  5 Abril 2024  
Nº Habitaciones:  2 Nº Adultos:  3  
165 € /Noche  
Reservar

Bhoolking: Registro

Nombre:   
Apellidos:   
DNI:   
Email:   
Comentarios:

Confirmo que tengo la mayoría de edad

Salir Confirmar

Ubicado en una playa soleada, este refugio playero invita al descanso con sus vistas al mar infinito. La brisa marina acaricia las ventanas, creando una sinfonía relajante. Un destino idílico para aquellos que buscan la serenidad frente al océano.

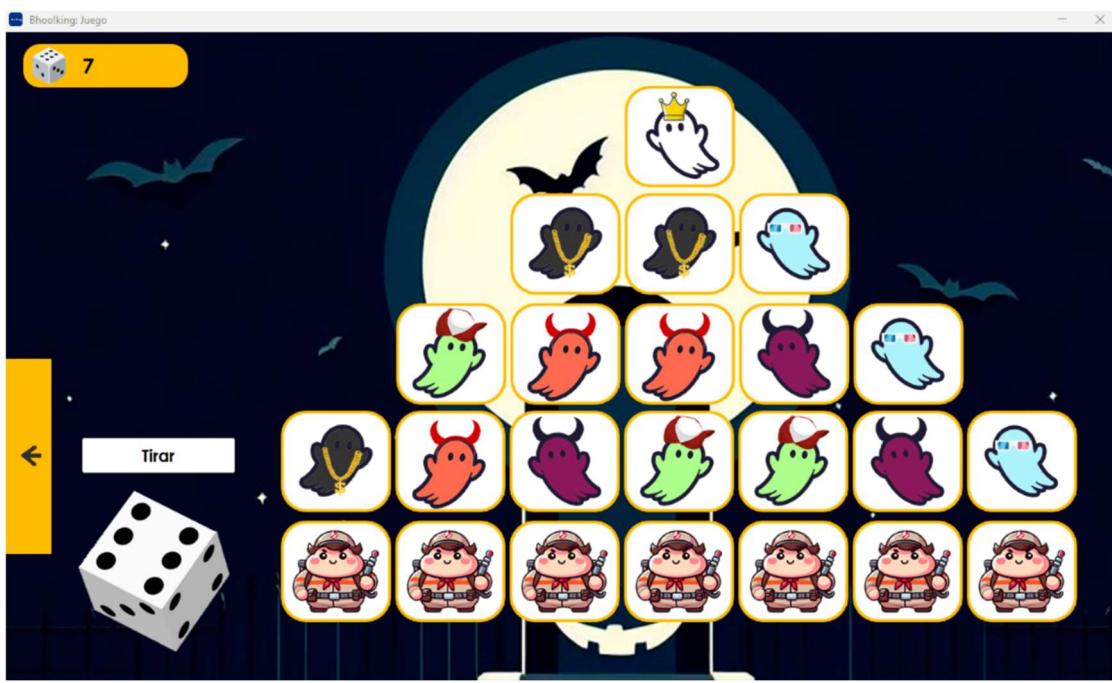
Autumn Lane 25, 1010, Grecia

En la pantalla de registro no se pide contraseña y tampoco se pregunta al usuario que si ya esta registrado. Solo queremos averiguar el dni para saber si el usuario tiene o no cupón. Tampoco se contemplaba la confirmación de la mayoría de edad.

## WIREFRAME

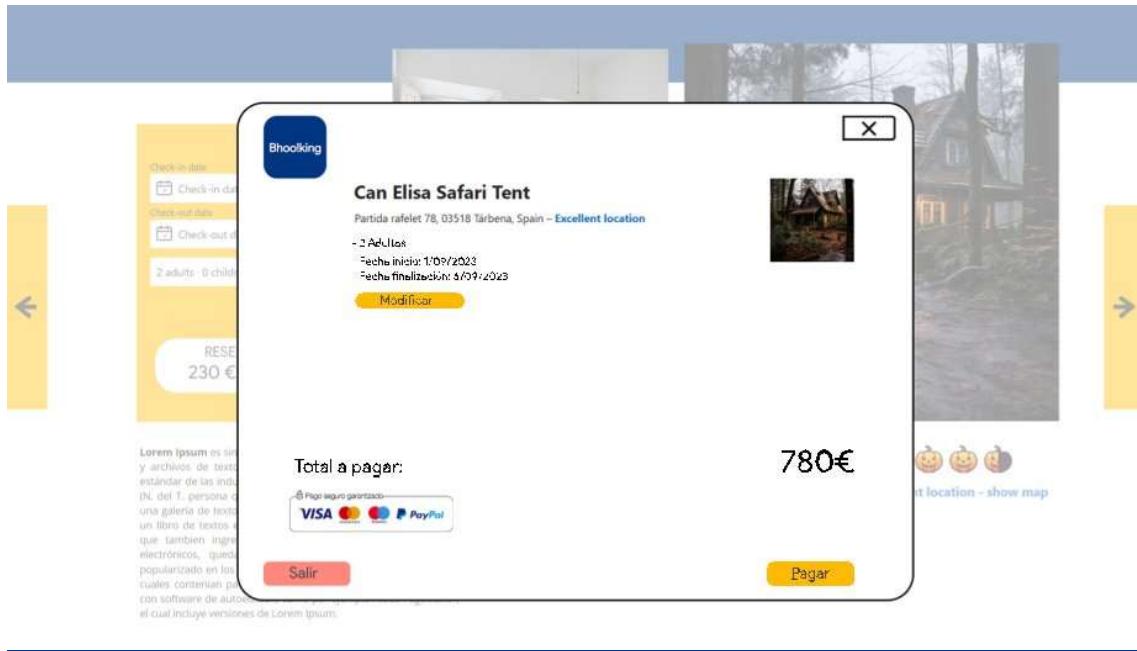


REAL

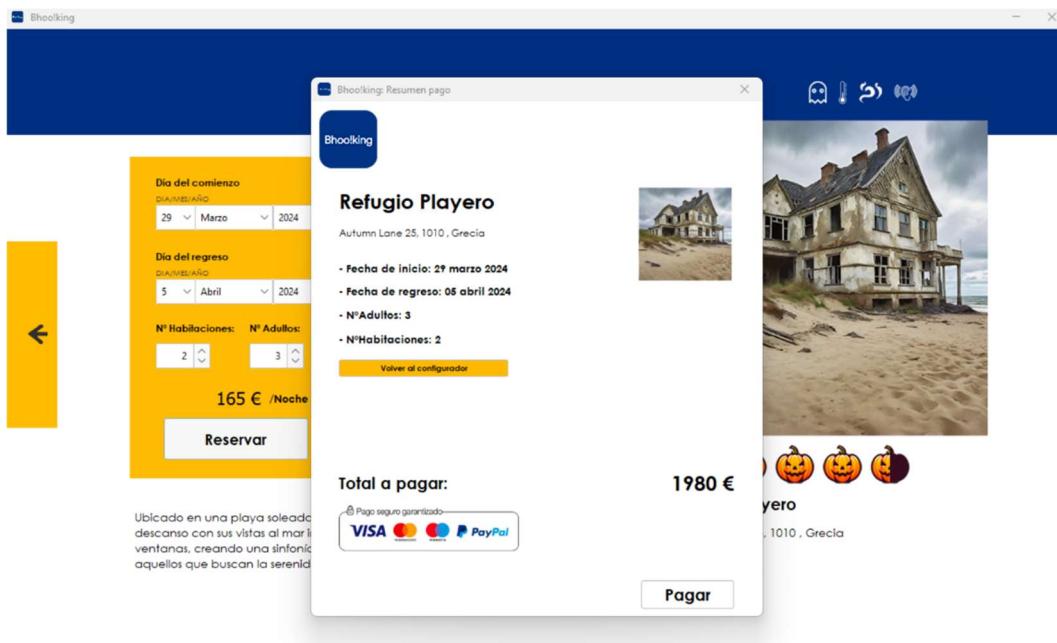


Con respecto al juego, el wireframe original no contemplaba los cazafantasmas. Además de estos, se ha añadido arriba a la izquierda un contador de intentos restantes.

## WIREFRAME



## REAL



La pasarela de pago de ha implementado casi idéntica, una pequeña mejoría fue en vez de llamar modificar, llamar "Volver al configurador"

## WIREFRAME



## REAL



En la página del cupón, no se ha implementado el confeti. Además, se muestra sobre la pantalla principal una vez finaliza el juego. También aparece el campo del DNI.

## Pruebas de usuario

Los resultados obtenidos tras la realización de los 7 escenarios utilizados para validar el Wireframe han sido esenciales, para determinar el diseño de la interfaz. Fácil para todos los usuarios e intuitivo para toda clase de públicos.

Se ha probado con usuarios finales y gracias a estos se han resuelto bugs, y se han añadido las siguientes funcionalidades dado el intento repetido por gran parte de estos:

- Para desplazarse en el carrusel no sirven solo los botones, sino las imágenes.
- También sirve clicar en el botón del dado para tirar además del botón

**Nombre:** Claudia

**Edad:** 26

**Ocupación:** Farmacéutica

**Escenario:** Claudia acaba de acabar la carrera y para celebrarlo va a hacer un viaje con sus amigas, como son muy aventureras y una de ellas vio en Instagram el anuncio de los castillos encantados han decidido ir. Como aún no han trabajado el juego de descuento les parece una buena idea; al ser jóvenes el juego se les da bien y obtienen el máximo descuento. No tiene ningún problema para hacer la reserva.

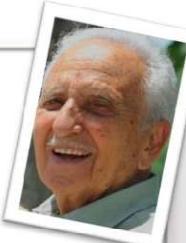


**Nombre:** Jesús

**Edad:** 80

**Ocupación:** Jubilado

**Escenario:** Jesús está paseando por la calle como de costumbre pero nota que hay una tienda nueva en el barrio y se adentra en ella. Es el local en el que se hacen las reservas a los castillos, no tiene pensado hacer ninguna reserva pero le llama la atención el juego ya que su nieta le está enseñando a manejar el móvil y estas cosas suponen un reto para él. Comienza a jugar y no consigue descuento pero como tiene tiempo vuelve a jugar y esta vez obtiene un 10% de descuento pero como cuando tiene que registrarse no sabe y deja la pantalla y se va.



**Nombre:** Marga

**Edad:** 45

**Ocupación:** Abogada

**Escenario:** Marga quiere regalarse a su pareja un viaje por su aniversario. Mientras espera a que le atiendan en la agencia de viajes, se acerca a un terminal en el que se anuncia una estancia inolvidable y la posibilidad de obtener descuentos en la reserva.

Ve claro por las explicaciones en el terminal que jugar no le compromete a reservar, así que prueba suerte. No obtiene ningún descuento, pero la aplicación le ofrece jugar de nuevo sin compromiso de reserva. Repite el juego y esta vez obtiene un descuento del 10%. La aplicación le deja claro que podría jugar de nuevo para obtener un descuento mayor, pero como ya le van a atender en el mostrador, prefiere irse y no guardar el descuento obtenido. Así, si no reserva un viaje tradicional, volverá para tratar de obtener el descuento máximo y reservar en un castillo encantado.



**Nombre:** Antonio

**Edad:** 79

**Ocupación:** Jubilado

**Escenario:** Antonio está apunto de cumplir 80 años y quiere celebrarlo con su familia con un viaje especial que recuerden todos. Se acerca a uno de los terminales de la agencia y ve claramente que antes de reservar le compensa jugar para tratar de conseguir un descuento. Tras leer las instrucciones básicas del juego, juega y gana un descuento del 25%. Le queda claro que es el mayor que puede conseguir, así que proporciona su DNI para usarlo después. Pasa a reservar y analiza las posibilidades: quiere ver los castillos que ofrecen todos los encantamientos menos los olores, que eso le da mucho asco. Encuentra uno que le encaja. Indica que va a reservar 5 habitaciones para 12 personas, pero el sistema le avisa que no es posible, así que añade una habitación más. Antonio indica que quiere utilizar el descuento obtenido. En el resumen de la reserva comprueba que está todo correcto, con el precio final ya actualizado, así que formaliza la reserva introduciendo los datos que le solicitan.



**Nombre:** Maya

**Edad:** 11

**Ocupación:** Estudiante

**Escenario:** Maya ha acompañado a su madre a una agencia de viajes, que tiene pensado reservar para pasar juntas unos días en Disneyland. Mientras espera, a Maya le llaman la atención los terminales que ofrecen jugar para obtener descuentos en una estancia muy especial. Juega varias veces hasta obtener el descuento máximo. Al pedirle el DNI para guardar el descuento, le indica claramente que ha de corresponder a una persona mayor de edad, así que no lo guarda. Por curiosidad, revisa la lista de hoteles disponibles y ve el tipo de experiencias que se ofrecen. Va corriendo a buscar a su madre para que no reserve en Disneyland si no que cambie el viaje para ir a uno de los castillos, que le mola mucho más.



