Eina web per generar avatars i utilitzar-los per generar productes decoratius personalitzats

Juan Antonio Font Foncubierta

28 de juny de 2025

Resum– En aquest projecte s'ha implementat una eina web per crear avatars personalitzats fent servir característiques facials predefinides. L'eina permet crear dissenys personalitzats fent servir textos, imatges de l'usuari i els avatars creats amb l'eina. Aquests elements es poden afegir al disseny i es poden moure, rotar i redimensionat amb el ratolí. També es poden moure cap al front o cap al fons per mostrar uns elements a sobre d'altres, i es poden eliminar del disseny si es desitja. Els dissenys es poden visualitzar a sobre de diferents elements decoratius com un quadre, una estora o un pòster blanc. L'eina pot enregistrar i identificar als seus usuaris i guardar els avatars i dissenys generats.

Paraules clau– Decoració, personalització avatars, dissenys, Vue, Express.js, SVG, imatges, MongoDB, CSS, HTML

Abstract– In this project we implemented a web app to create personalized avatars using predefined facial elements. The web App allows the creation of personalized designs using text, images and the avatars created with the app. This elements can be added to the design and can be moved, rotated and resized using the mouse. They can also be moved to the front or to the back to show elements on top of each other.They can also be removed from the design if desired. The designs can be viewed on top of diferent decorative elements such as a painting, a doormat or a blank poster. The web app can register and identify users and saves their avatars and dessings

Keywords– Decoration, personalization, avatars, desings, Vue, Express.js, SVG, images, MongoDB, CSS, HTML

1 Introducció

La tecnologia s'ha convertit en un aspecte clau a la nostra vida quotidiana. En l'àmbit de la personalització d'objectes, existeixen moltes pàgines web que ofereixen objectes com tasses, espelmes o quadres personalitzats amb diferents elements com per exemple, una fotografia amb algun amic o familiar o un text de felicitació.

En aquest treball volem implementar una eina web que ofereixi un servei de personalització d'objectes. En la secció d'estat de l'art veurem diferents eines existents i les analitzarem per veure com és el seu servei de personalització i establirem uns objectius basant-nos en aquesta anàlisi. Tot seguit exposarem la metodologia i la planificació que s'ha fet servir per dur a terme el projecte i exposarem els resultats obtinguts explicant en detall les funcionalitats del nostre projecte. I finalment extraurem unes petites conclusions sobre el projecte.

• E-mail de contacte: JuanAntonio.Font@gmail.com
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Marc Ortega Gil
• Curs 2024/25

2 Estat de l'art

Abans de començar a implementar el nostre projecte, hem buscat a la xarxa eines web que ofereixin característiques similars al que volem implementar nosaltres. Existeixen nombroses pàgines web que ofereixen productes amb dissenys personalitzats, la gran majoria d'aquestes pàgines treballen en el mercat de regals personalitzats i et perme-

Fig. 1: Editor de dissenys de Regalo Original

ten demanar entre una gran varietat d'objectes com tasses, miralls, espelmes i fins i tot retalls de cartó a mida real d'una persona. Tots aquests objectes es poden personalitzar amb diferents imatges i textos que l'usuari pot escollir.

Una d'aquestes pàgines que hem trobat notable és Regalo Original[1], aquesta empresa es dedica a la venda d'un gran assortiment d'objectes per fer regals tant a amics com a familiars, i gran part del seu catàleg es pot personalitzar amb imatges, similar al que volem fer nosaltres. El seu editor de dissenys té les següents funcionalitats:

- Afegir una imatge del nostre ordinador
- Afegir text i canviar-ne el color i la font.
- Escollir el color de fons de tot el disseny.
- Moure, rotar i redimensionar, tant les imatges com el text, amb el ratolí.

Partint de l'editor de dissenys de Regalo Original(Fig. 1) com a exemple, volem afegir la possibilitat de poder crear avatars personalitzats que puguin representar als amics o familiars a qui faríem aquest regal, i integrar aquesta nova funcionalitat amb la creació del disseny personalitzat.

3 Objectius

En aquest projecte es proposa fer una eina web que permeti la creació d'avatars i dissenys personalitzats. Els avatars es crearan a partir d'una selecció de diferents característiques facials ja predefinides entre les quals es pot escollir. Els dissenys es podran personalitzar afegint text, imatges proporcionades per l'usuari i els avatars creats amb aquesta mateixa eina. Basant-nos en el que hem vist a l'estat de l'art, ens hem marcat els següents objectius inicials:

- Creació d'avatars personalitzats fent servir característiques facials predefinides.
- Crear dissenys personalitzats amb diferents elements
  - Text proporcionat per l'usuari dins de la web
  - Imatges de l'usuari emmagatzemades al seu dispositiu.
  - Avatars creats amb aquesta mateixa eina.
- Visualitzar els dissenys en un element decoratiu
- Identificar usuaris per poder guardar i recuperar els seus dissenys i avatars.

L'enfocament principal d'aquest projecte és crear una eina funcional que integri totes aquestes característiques en una única pàgina web, per tant, per estalviar temps, l'estil de la pàgina web no serà una prioritat. Per aquesta mateixa raó, tampoc es considerarà un objectiu el disseny i l'estil artístic dels avatars.

3.1 Requisits

Després de veure els objectius generals del projecte, definirem amb més detall els requisits de la nostra eina.

- En el creador d'avatars hi haurà cinc categories: sexe, ulls, cabells, llavis i celles, cadascuna amb dues opcions per escollir.
- Els canvis en l'avatar es visualitzaran de forma dinàmica.
- En l'editor de dissenys, es podran afegir, moure, rotar o redimensionar els elements (text, imatges, i/o avatars) dins del disseny.
- Es podrà desfer un disseny per començar de zero o carregar un disseny predefinit.
- El visualitzador de decoracions disposarà de tres elements decoratius per escollir: un quadre, una estora o un pòster blanc.

4 Metodologia

Per dur a terme aquest projecte s'ha optat per seguir una metodologia iterativa. S'ha escollit aquesta metodologia per poder realitzar el projecte en diferents iteracions on es treballarà una de les funcionalitats del projecte en cada iteració. Això en permetrà construir el projecte afegint característiques de forma gradual.

Per realitzar la implementació d'aquest projecte hem fet servir Vue.js [2], un framework de JavaScript[3] que permet crear les interfícies d'usuari en una Single Page Application[4] (SPA). Per la gestió dels usuaris de l'aplicació i les seves dades, hem implementat un backend amb Express.js[5] que gestiona la base de dades, i s'encarrega de registrar i verificar als usuaris, així com guardar els avatars i dissenys en fitxers.

La pagina web de l'eina està dividida en 4 seccions: Avatars, Dissenys, Decoracions i Login. Cada secció està formada per un component Vue principal que conté la vista de la secció i altres components Vue que implementen la lògica de la secció. La navegació per les diferents seccions de l'eina s'ha implementat fent servir la llibreria nativa d'encaminament de Vue, anomenada Vue-Router[6]. Aquesta llibreria proporciona un objecte Router al qual li hem d'especificar les subpàgines que conté la nostra eina i quin component Vue es mostrarà en aquella subpàgina.

Per a la base de dades s'ha fet servir MongoDB[7], una base de dades noSQL que emmagatzema dades en estructures de clau-valor que anomena Documents[8]. Per interactuar amb aquesta base de dades hem fet servir la llibreria de MongoDB per Node.js, mongodb[9].

La base de dades emmagatzema la informació sobre els usuaris en una única col·lecció que conté a tots els usuaris identificats pel seu correu electrònic. Els dissenys i avatars guardats pels usuaris es troben a un directori a l'arrel del projecte, identificats amb un Identificador Universal Únic[10] (UUID), que també es guarda a la base de dades.

5 Planificació

Com s'ha mencionat a la metodologia, el projecte es dividirà en iteracions, a on cada iteració es treballarà una de les funcionalitats del projecte. Per aquest motiu, s'ha decidit dividir aquestes funcionalitats en tasques. La primera tasca que es realitzarà serà una primera fase del disseny de les interfícies d'usuari per definir com es veuran les diferents seccions de la pàgina web. Aquesta tasca tindrà una duració de dues setmana. Després d'aquesta primera tasca, les següents tasques tindran una durada de tres setmanes i cadascuna d'elles correspondrà a una de les característiques de l'eina web. I per finalitzar es realitzaran la redacció de l'informe final i la preparació de la presentació de la defensa pública.

L'ordre de realització de les tasques serà el següent:

- Editor d'Avatars sense la funcionalitat de guardar
- Editor de dissenys sense la funcionalitat d'afegir avatars ni de guardar dissenys
- Visualitzador de decoracions
- Sistema de gestió d'usuaris juntament amb la integració de les altres característiques
- Redacció de l'informe final i preparació de la presentació pública.

A la figura 2 es pot veure el diagrama de Gantt del projecte.

Hem decidit escollir aquest ordre de realització per afavorir que les funcionalitats de l'eina web siguin el més modulars possibles, ja que la integració serà l'últim pas, implementarem les altres funcionalitats de tal forma que siguin fàcilment integrables entre elles en aquesta última fase.

6 Resultats

En les següents seccions exposarem totes les funcionalitats que s'han implementat per assolir tots els objectius proposats.

7 Avatars

La primera funcionalitat implementada ha estat el creador d'avatars. Aquesta secció de l'eina permet als usuaris generar un avatar seleccionant característiques facials predefinides en cinc categories diferents: Cabells, Ulls, Llavis, Celles i Sexe. El component Vue principal de la secció conté cinc botons, un per cada categoria, i dos components Vue encarregats de gestionar els menús de les categories i la vista de l'avatar que l'usuari està creant.

El primer subcomponent, ControllerMenu, rep del component principal, mitjançant un prop[11], el botó que ha pres l'usuari i s'encarrega de mostrar el component Menu que correspon a la categoria seleccionada per l'usuari. També manté el registre de les seleccions de l'usuari i informa al component principal d'aquestes seleccions mitjançant un event quan hi ha un canvi. Cadascun dels diferents components Menu mostra les diferents opcions disponibles per aquella categoria en forma d'imatge. Són aquests components els que s'encarreguen d'enregistrar la selecció de l'usuari quan es fa clic a sobre d'una de les imatges, i informen al component ControllerMenu mitjançant un event quan hi ha un canvi en la selecció. A la figura 3 es pot veure el menú Celles desplegat per l'usuari, mostrant els elements disponibles.

El segon component, ControllerAvatarSvg, utilitza la llista de seleccions proporcionades pel component principal per mostrar un SVG (Fig. 4) amb una vista prèvia de l'avatar que l'usuari està creant. Aquesta vista s'actualitza dinàmicament amb els canvis en les seleccions de l'usuari. L'element SVG consta de cinc imatges, una per categoria, que se superposen per mostrar un avatar sencer. El camí on es troba cada imatge per mostrar es computa a partir de la llista de seleccions i es guarda en variables computades de Vue, les quals és recomputen i actualitzen el seu valor automàticament cada vegada que canvia la selecció de l'usuari.

Si l'usuari s'ha identificat, pot prémer el botó per guardar el seu disseny. Per guardar-lo, s'envia al backend una petició POST amb la llista de seleccions actual en format JSON i el token que identifica l'usuari.

Quan el servidor rep aquesta petició, primer es fa servir una peça de middleware que s'encarrega de parsejar al petició per extreure el JSON del cos de la petició. El backend genera un SVG similar al que es mostra en la pàgina web, però substitueix les imatges per les dades SVG corresponents. Això ho fem per a que la imatge de l'avatar es pugui visualitzar sense la necessitat de fer peticions al servidor per a que ens serveixi les imatges dels elements facials.

Una vegada construït el SVG, es genera un UUID aleatori que serveix com a identificador de l'avatar, es guarda en un fitxer posant aquest identificador com a nom del fitxer i s'afegeix aquest identificador a llista d'identificadors d'avatar de l'usuari. A la figura 5 podem trobar un exemple d'un avatar generat amb aquesta funcionalitat.

8 Dissenys

Aquesta és la funcionalitat principal de la nostra eina web. En aquesta secció de la pàgina es mostra un element SVG buit amb un fons gris, que l'usuari farà servir com a llenç en blanc per crear el seu disseny. Al costat de l'element SVG es mostren diferents botons, cadascun corresponent a una acció que es pot fer al disseny.

Per començar, si l'usuari s'ha identificat, pot afegir el seu avatar creat amb la funcionalitat anterior. Quan es prem el botó per afegir avatars es llencen una sèrie de peticions al backend fent servir la funció fetch[12], una

	Març				Abril				Maig				Juny			
	10	17	24	31	7	14	21	28	5	12	19	26	2	9	16	23
Diseny de les interficies																
Implementació de l'editor d'avatars																
Implementació de l'editor de disenys																
Implmenetació del visualitzador de decoracions																
Implementació de login																
Preparacio Informe final																
Preparació presentació defensa																

Fig. 2: Diagrama de Gantt del projecte

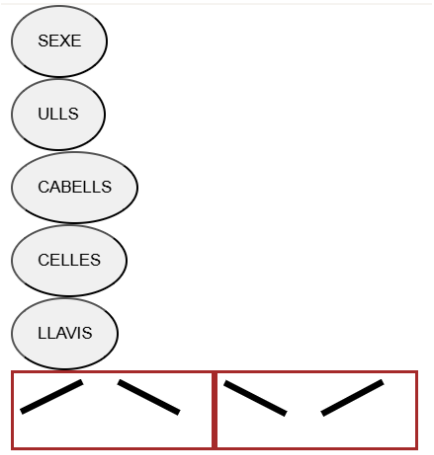

Fig. 3: Botons de categories amb el menu Celles desplegat

```
<svg width="512" height="512" xmlns="http://www.w3.org/2000/svg">

  <image :href="sexePath" x="136" y="49" height="500" width="250"></image>
  <image :href="cellesPath" x="178" y="91" height="32" width="150"></image>
  <image :href="ullsPath" x="181" y="126" height="32" width="150"></image>
  <image :href="llavisPath" x="195" y="210" height="32" width="150"></image>
  <image :href="cabellsPath" x="115" y="28" height="300" width="300"></image>

</svg>
```

Fig. 4: Element SVG

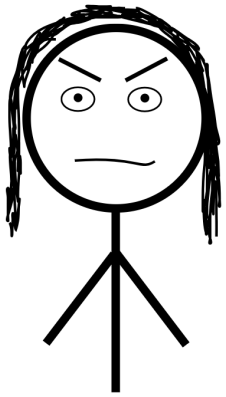

Fig. 5: Exemple d'avatar

primera petició per recuperar una llista de UUID dels avatars associats a l'usuari i per cada UUID rebut, es fa una petició per recuperar el fitxer de l'avatar corresponent.

Els avatars recuperats es mostren per pantalla en un menú que se superposa a la pàgina (Fig. 6) i l'usuari pot seleccionar l'avatar que desitgi fent clic a sobre. Quan se selecciona un avatar, les dades en format SVG és parsejen per eliminar els salts de línia i els espais en blanc i se substitueixen totes les cometes dobles per cometes simples. Tot seguit les dades es codifiquen en un URI de dades amb el prefix 'data:image/svg+xml' i s'encapsula en un element imatge per afegir-lo a l'element SVG.

Per afegir text, es fa servir un Text input HTML on l'usuari pot introduir el text que desitgi. Quan es prem el botó d'afegir text, les dades de l'input es copien a un element Text, que s'afegeix al SVG per ser renderitzat.

L'usuari pot afegir les seves pròpies imatges fent servir l'element File input HTML per seleccionar una imatge de la seva màquina local. Quan es prem el botó per afegir la imatge es crea un objecte FileReader[13] que llegeix el fitxer proporcionat per l'usuari. En aquest cas li especifiquem a l'objecte FileReader que volem que ens retorni el fitxer com a un URI de dades fent servir el mètode readDataAsURL, per poder afegir-lo al SVG.

No ens cal detectar el format del fitxer, ja que FileReader ho farà de forma automàtica i afegirà el prefix corresponent a les dades. Quan obtenim l'URI de dades, que conté tant el format de la imatge com les dades de la imatge codificades en base 64[14], creem un element image fent servir aquest URI de dades i l'afegim a l'element SVG.

Una vegada hem afegit els elements que vulguem, podem interactuar amb ells amb el ratolí per moure'ls, rotar-los i redimensionar-los. Per implementar aquestes funcionalitats hem fet servir les llibreries Vue3-Moveable[15] i Vue3-Selecto[16]. La llibreria Selecto s'encarrega de gestionar la selecció dels diferents elements del SVG quan es fa clic a sobre d'ells, i també s'encarrega d'indicar a Moveable quin element ha de modificar. La llibreria Moveable aplica a l'element seleccionat les transformacions afins pertinents per tal de moure, rotar o redimensionar l'element.

Aquestes llibreries funcionen afegint eventListeners a l'element seleccionat, i modificant l'atribut CSS transform[17] en funció de com s'interactua amb l'element. Per moure l'element s'afegeix la funció CSS translate[18] que mou l'element a unes coordenades x, y especificades. Quan es vol rotar l'element, es fa servir la

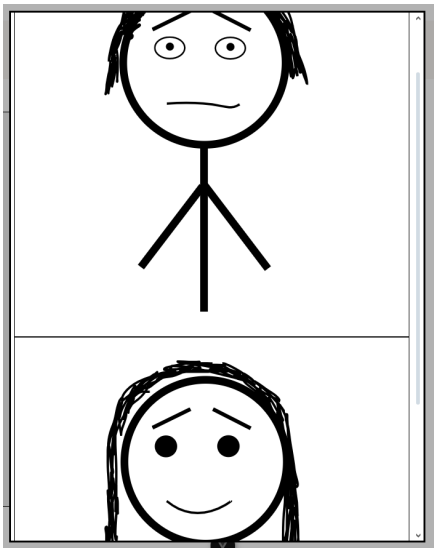

Fig. 6: Menú sobreposat mostrant els avatars de l'usuari

funció rotate[19], la qual rota l'element seleccionat l'angle que se li especificaa al voltant del seu centre, i per redimensionar l'element es fa servir la funció scale[20], que rep un parell de coordenades x, y que fa servir com a vector per escalar l'element.

La selecció d'un element es determina fent servir els events 'click' dels elements dins del SVG. La llibreria Selecto escolta aquests events i manté una variable, que anomenem target, que conté un objecte ref[21], que és una referencia a l'ultim objecte que s'ha fet clic. La llibreria Moveable pot fes servir aquesta variable target per accedir a l'element seleccionat i modificar-lo.

Per a que les transformacions es renderitzin de manera correcta, tots els elements afegits al disseny tenen els seus atributs d'estil especificats de fomra explicita amb l'atribut HTML style[22], això ens permet renderitzar els dissenys sense la necessitat d'una fulla d'estil CSS separada. L'estil que s'afegeix als elements és el següent:

- transform-box: fill-box. Indica que el punt d'origen de les transformacions és relatiu a l'element, i no a l'element pare.
- transform-origin: center. Indica que el punt d'origen de les transformacions és el centre de l'element.

Per indicar a les llibreries sobre quins elements volem que treballin, fem servir les classes de CSS. Hem d'especificar una àrea que contindrà elements seleccionables i quins dels elements són o no seleccionables. El nostre element SVG serà la nostra àrea seleccionable, li donarem la classe 'selecto-area' i els elements que conté, menys el rectangle gris que actua com a fons, seran de la classe 'selectable'. Per tant indiquem a Selecto que ha de treballar sobre els elements de classe 'selecto-area' i 'selectable'. L'element seleccionat es guarda en una variable que Selecto modifica amb una referencia a l'ultim element que s'ha fet clic. La llibreria Moveable fa servir aquesta variable per accedir a l'element seleccionat i aplicar les modificacions a la posició, rotació i escala de l'element.

L'element seleccionat pot ser mogut cap al front o cap al fons amb respecte dels altres elements per tal que aparegui a sobre o a sota dels altres elements. I també pot

```
down(){
  let svg = document.getElementById("svg")
  let children = svg.childNodes

  children.forEach((element,index) => {
    if (element == this.lastTarget[0]){
      //Evita mover el elemento detras del background
      if (index != 1) {
        svg.insertBefore(element, children[index-1])
      }

    }
  });
  //Mantiene el elemento seleccionado
  this.targets = this.lastTarget
},
```

Fig. 7: Funció per moure un element cap al fons

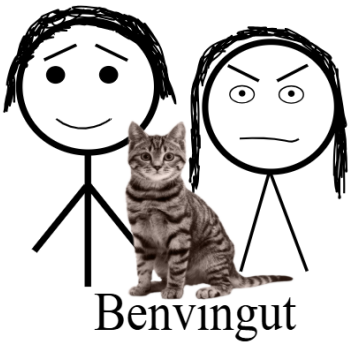

Fig. 8: Exemple de disseny

ser eliminat del disseny. Per aconseguir això es reordenen els elements dins de la llista d'element SVG. La llista d'elements es renderitza per ordre, això implica que els primers elements apareixen per sota dels últims. En el nostre cas, quan es prem el botó per moure els elements cap al front o fons, l'element seleccionat intercanvia la seva posició en la llista amb l'element posterior o anterior, sempre asegurant-nos que no el movem darrere del fons.

Similar a quan es guarda un Avatar, per guardar un disseny fem una petició al servidor on enviem el token de l'usuari i al cos de la petició posem l'element SVG que conté el disseny de l'usuari. El backend guardarà l'element en un fitxer, i associarà el UUID generat als dissenys de l'usuari. A la figura 8 es pot trobar un exemple de disseny generat amb dos avatars generats per l'usuari, una imatge i un element de text.

9 DECORACIÓ

En aquesta secció de la pàgina l'usuari pot seleccionar un dels seus dissenys i afegir-lo a un element decoratiu. L'usuari ha d'haver identificat primer per poder afegir els seus dissenys.

Aquesta pàgina mostra un element SVG que conté la imatge d'un element decoratiu buit. Aquest element pot ser el marc d'un quadre, una estora o un pòster en blanc. Es pot canviar entre aquests tres elements amb el botó de selecció de decoració.

Quan es prem el botó per afegir un disseny, el llença una petició al servidor per recuperar els identificadors de dissenys associats a aquest usuari i, igual que per recupe-

rar els avatars, per cada identificador es llença una petició per recuperar el fitxer que conté el disseny.

Després d'obtenir tots els dissenys de l'usuari, es mostren en un menú sobreposat a la pàgina i es pot seleccionar un d'ells fent clic a sobre per afegir-lo a l'element SVG. Una vegada afegit, fem servir una altra vegada la llibreria Moveable per poder interactuar amb el disseny amb el ratolí per ajustar-lo a l'element decoratiu.

Per poder interactuar amb el disseny en tot moment, sense que la imatge de fons interfereixi amb les nostres accions amb el ratolí, hem d'especificar que les imatges dels elements decoratius tinguin la propietat CSS 'pointer-events: null'. Això deshabilita els seus events i ens permet fer clic a través de les imatges.

10 USUARIS

La gestió dels usuaris és una part important d'aquest projecte, ja que ens permet integrar totes les altres funcionalitats de l'eina. A la base de dades, els usuaris es guarden en un Document de MongoDB amb els següents camps:

- _id: El correu electrònic de l'usuari. Es fa servir com a identificador únic d'aquest usuari.
- name: El nom de l'usuari.
- passw: El hash i el salt de la contrasenya de l'usuari.
- avatarIds: Llista d'identificadors d'avatar associats a l'usuari.
- dissenysIds: Llista d'identificadors de dissenys associats a l'usuari.

Per guardar les contrasenyes dels usuaris fem servir la llibreria crypto[23] de JavaScript per generar un salt i fer el hash de la contrasenya amb aquest salt. Per generar el hash fem servir l'algorisme SHA256 amb 1000 iteracions, això dificulta els atacs de força bruta per revertir el hash. A la figura 9 es mostra un exemple de les dades d'un usuari.

El salt consta de setze bytes generats aleatòriament amb la funció randomBytes de la llibreria crypto, i la seva funció és evitar que dues contrasenyes idèntiques tinguin el mateix hash i així poder evitar atacs amb diccionaris.

Com que fem servir el correu electrònic de l'usuari com al seu identificador, ens hem d'assegurar que dos usuaris no puguin fer servir el mateix correu per registrar-se. Per aconseguir això, ens aprofitem del fet que la base de dades MongoDB no permet afegir dos elements a la mateixa col·lecció que tinguin un camp _id idèntic. Quan intentem afegir un _id que ja existeix, ens retorna l'error Duplicate Key. Quan el backend es troba amb aquest error en el registre de l'usuari, es cancel·la el registre i retorna a la petició de l'usuari el codi d'estat 409 Confict, per indicar que ha de fer servir un correu electrònic diferent.

En la secció de login de la pàgina web es troben els formularis per iniciar sessió (Fig. 10) i per crear un compte (Fig. 11).

Quan un usuari inicia sessió de forma satisfactòria, el backend li respon amb un objecte JSON que conté el seu nom i un token de sessió. Aquesta informació es guarda al navegador de l'usuari fins que es tanca la sessió o l'usuari tanca la finestra o el navegador, cosa que esborra el token. Per guardar la informació fem servir l'API de sessionStorage[24]. Sempre que l'eina llença una petició al backend per recuperar els identificadors d'avatars o de dissenys es comprova si aquest token existeix a la memòria del navegador i, si es així, s'afegeix a la capçalera Authorisation de la petició. Si aquesta capçalera no es troba a la petició, el backend respondrà amb un codi d'estat 401 Unauthorised. En el backend es manté una llista de tokens generats i per a quin usuari s'ha generat aquell token.

Per guardar els dissenys i avatars, fem servir la funció writeFileSync[25] de la llibreria fs[26] de Javascript, on especifiquem l'identificador de l'avatar o disseny com a nom del fitxer. Per després recuperar els dissenys o avatars de l'usuari, hem de fer una consulta a la base de dades per saber els identificadors associats a l'usuari. Amb els identificadors podem trobar els fitxers associats a l'usuari al directori userFiles a l'arrel del projecte.

11 LINIES DE CONTINUACIÓ

En aquesta secció es proposen diferents millores per tal d'ampliar les funcionalitats d'aquesta eina.

11.1 Ampliació dels avatars

El creador d'avatars es podria ampliar afegint més categories per escollir, com per exemple nas, orelles o galtes. També es podria afegir la funcionalitat de modificar la mida i la posició dels elements facials, així com algunes característiques com l'espai entre els ulls, la curvatura dels llavis o la inclinació de les celles. Per ampliar encara més la personalització dels avatars es proposa poder afegir diferents accessoris com gorres o ulleres.

11.2 Mes elements en els dissenys

El SVG de l'editor de dissenys no està limitat a només text, imatges i avatars. Es proposa afegir més funcionalitats a aquest editor per a que permeti afegir formes geomètriques com rectangles, cercles o triangles. També es podria afegir la possibilitat de modificar el color dels diferents elements, així com la font del text. També es podria afegir la funcionalitat de dibuixar a mà alçada, per permetre a l'usuari personalitzar encara mes els seus dissenys.

11.3 Publicació de la web

A llarg termini, es proposa preparar la web per a la seva publicació. Això inclou revisar els aspectes de seguretat de la web, adquirir un nom de domini i un servidor per allotjar la web, a més de crear un certificat digital del nom de domini per poder fer servir el protocol HTTPS. De cara al públic, s'hauria de fer proves d'experiència d'usuari (UX) per millorar i validar la usabilitat de la web.

També es podria començar un negoci de venda de productes decoratius personalitzats, implementant un sistema de pagament i enviaments a domicili.

```
{
  _id: 'testUser@testDomain.com',
  name: 'Test User',
  passw: {
    hash: '9335abdff5a5dbda1491b980a92f8052c5a84d6ddae0793c75e87f6fcbecedd04d6301bbf3855dba74f662fe8f77e369c37881fcc27fbd8e687e36ca59b78fbc',
    salt: '49a11f87c89e3da99fc97ed14c9acdc2'
  },
  avatarIds: [ '3934c3b2-c457-46f7-a388-d440e9d03053' ],
  dissenysIds: [ '2fc55394-f5b9-497e-a7d2-8e1526908d66' ]
}
```


Fig. 9: Dades d'un usuari de prova

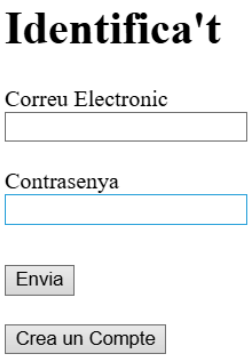

Fig. 10: Formulari per iniciar sessió

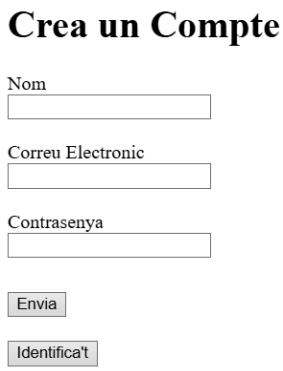

Fig. 11: Formulari per crear un compte

12 Conclusions

L'objectiu d'aquest projecte era la creació d'una eina web que permetis la creació de dissenys personalitzats fent servir diferents elements com text o imatges, a mes, volíem donar la possibilitat als usuaris d'expandir les seves opcions de personalització creant els seus avatars personalitzats per afegir als dissenys. I també permetre veure com quedaria aquest disseny sobre un element decoratiu. En retrospectiva, podem dir que hem assolit tots els objectius de forma satisfactòria.

Per crear el nostre projecte, hem fet servir Vue.js, juntament amb les llibreries Moveable i Selecto per fer les interfícies d'usuari i implementar les funcionalitats per crear avatars i dissenys. Per gestionar els usuaris i fer la gestió dels dissenys i avatars guardats, hem creat un servidor web amb Express.js que es comunica amb la nostra aplicació de Vue i fa servir les llibreries crypto per la generació de contrasenyes, mongodb per la comunicació amb la base de dades i la llibreria fs per guardar els avatars i dissenys en arxius. Per guardar els usuaris hem fet servir una base de dades de MongoDB.

Durant el desenvolupament del projecte van sorgir diferents dificultats, com per exemple, la incorrecta renderització del SVG a l'hora de guardar-los que provocava que els textos i imatges es moguessin fora de vista, que es va solucionar afegint els estils explícits. Un altre problema que va sorgir va ser que a l'hora d'afegir imatges amb la funció appendChild[27] no s'actualitzava el SVG, això es va solucionar fent la concatenació del text HTML de la imatge amb el del SVG, que provoca que es renderitzi el SVG un altra vegada i mostri la imatge. Per sort, vam poder solucionar aquests inconvenients de forma ràpida i no ha calgut fer una modificació de la planificació.

Agraïments

Agraïments al professor i tutor d'aquest projecte Marc Ortega Gil pel seu ajut i suport durant el desenvolupament d'aquest treball al llarg d'aquests mesos.

Referències

[1] ≪Regalos Originales - Desayunos a Domicilio - Regalo Original≫. Consulta: 3 març 2025. [En línia]. Disponible a: https://www.regalooriginal.com/

[2] ≪Vue.js≫. Consulta: 3 març 2025. [En línia]. Disponible a: https://vuejs.org/

[3] ≪JavaScript — MDN≫. Consulta: 3 març 2025. [En línia]. Disponi-

ble a: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[4] ≪SPA (Single-page application) - Glossary — MDN≫. Consulta: 8 març 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Glossary/SPA

[5] ≪Express - Node.js web application framework≫. Consulta: 3 març 2025. [En línia]. Disponible a: https://expressjs.com/

[6] ≪Vue Router — The official Router for Vue.js≫. Consulta: 8 març 2025. [En línia]. Disponible a: https://router.vuejs.org

[7] ≪MongoDB: The World's Leading Modern Database≫, MongoDB. Consulta: 8 març 2025. [En línia]. Disponible a: https://www.mongodb.com/

[8] ≪Documents - Database Manual - MongoDB Docs≫. Consulta: 18 juny 2025. [En línia]. Disponible a: https://www.mongodb.com/docs/manual/core/document/

[9] ≪MongoDB Node Driver - Node.js Driver v6.17 - MongoDB Docs≫. Consulta: 18 juny 2025. [En línia]. Disponible a: https://www.mongodb.com/docs/drivers/node/current/

[10] P. J. Leach, R. Salz, i M. H. Mealling, ≪A Universally Unique IDentifier (UUID) URN Namespace≫, Internet Engineering Task Force, Request for Comments RFC 4122, jul. 2005. doi: 10.17487/RFC4122.

[11] ≪Props≫. Consulta: 18 juny 2025. [En línia]. Disponible a: https://vuejs.org/guide/components/props.html

[12] ≪Fetch API - Web APIs — MDN≫. Consulta: 18 juny 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

[13] ≪FileReader - Web APIs — MDN≫. Consulta: 18 juny 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/API/FileReader

[14] ≪Base64 - Glossary — MDN≫. Consulta: 18 juny 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Glossary/Base64

[15] D. (Younkue Choi), daybrush/moveable. (19 maig 2025). TypeScript. Consulta: 18 juny 2025. [En línia]. Disponible a: https://github.com/daybrush/moveable

[16] D. (Younkue Choi), daybrush/selecto. (19 maig 2025). TypeScript. Consulta: 18 juny 2025. [En línia]. Disponible a: https://github.com/daybrush/selecto

[17] ≪transform - CSS — MDN≫. Consulta: 19 maig 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/CSS/transform

[18] ≪translate() - CSS — MDN≫. Consulta: 19 maig 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/translate

[19] ≪rotate() - CSS — MDN≫. Consulta: 19 maig 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/rotate

[20] ≪scale() - CSS — MDN≫. Consulta: 19 maig 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function/scale

[21] ≪Ref≫. Consulta: 19 maig 2025. [En línia]. Disponible a: https://vuejs.org/api/reactivity-core.html#ref

[22] ≪HTML style global attribute - HTML — MDN≫. Consulta: 21 maig 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Global_attributes/style

[23] ≪Crypto — Node.js v24.2.0 Documentation≫. Consulta: 3 juny 2025. [En línia]. Disponible a: https://nodejs.org/api/crypto.html

[24] ≪Window: sessionStorage property - Web APIs — MDN≫. Consulta: 3 juny 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage

[25] ≪writeFileSync — Node.js v24.2.0 Documentation≫. Consulta: 3 juny 2025. [En línia]. Disponible a: https://nodejs.org/api/fs.html#fswritefilesyncfile-data-options

[26] ≪File system — Node.js v24.2.0 Documentation≫. Consulta: 3 juny 2025. [En línia]. Disponible a: https://nodejs.org/api/fs.html#file-system

[27] ≪Node: appendChild() method - Web APIs — MDN≫. Consulta: 3 juny 2025. [En línia]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/API/Node/appendChild