DAWC

Adrià Navarro Savall

CRUD **3**
Explicació HTML Categories **3**
CategoriesNoticies 3
CategoriaLlista 4
Explicació JS Categories **5**
CategoriesNoticies 5
CategoriaLlista 8
Explicació HTML Usuari **12**
Usuari 12
editarUsuari 13
Explicació JS Usuari **14**
Usuari 14
editarUsuari 18

CRUD

Aquest és un CRUD sobre la taula **categories**. Aquest consisteix en la creació, lectura, modificació i eliminació de la pròpia taula. El propòsit d'aquest codi és crear les categories, editar-les una vegada creades complint certs requisits, llegir les categories, modificar-les respectant els criteris mínims i també poder eliminar les categories que siguen necessàries.

Explicació HTML Categories

CategoriesNoticies

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulary Noticies</title>
    <script type="text/javascript" src="CategoriesNoticies.js"></script>
</head>
<body>
    <h1>Categoria per la noticia</h1>
    <form>
        <label>Categoria</label>
        <br><br>
        <input type="text" placeholder="Categoria de la  noticia" id="category" maxlength="20" pattern="[A-Z][a-z]{1,19}" title="Introdueix de 2 a 20 lletres i començant amb Majú
        <br><br>
        <button type="button" id="btSend">Enviar</button>
        <button type="button" id="btList">LListar Categories</button>
    </form>

    <p id="message"></p>
    <div id="categoryList"></div>
</body>
</html>
```

Aquest és el HTML. Aquest arxiu té com a propòsit mostrar a l'usuari les opcions que té a l'hora d'utilitzar-lo. En aquest cas, mostra per pantalla un input on podrà introduir el nom de les categories, després hi ha dos botons: un per afegir la categoria i poder guardar-la, i un altre per poder veure la llista de categories.

- btSend: Enviar les dades al nostre
- btList: Ens portara a la pagina per veure el llistat de categories
- Input: Ací tenim una validació per a que la paraula siga mínim 2 i màxim 20 lletres, i que sempre tinga que començar per una majúscula.
- message: En cas d'introduir una categoria, ens mostrarà un missatge on ens dirà si la categoria és correcta o no.

CategoriaLlista

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulary Noticies</title>
    <script type="text/javascript" src="CategoriesLlistat.js"></script>
</head>
<body>
    <h1>Categoria per la noticia</h1>
    <form>
        <label>Categoria</label>

        <button type="button" id="btList">LListar Categories</button>
        <button type="button" id="btInput">Afegir Categoria</button>
    </form>

    <div id="categoryList"></div>
</body>
</html>
```

Aquesta és la segona part del HTML de categories. Aquesta s'encarregarà de mostrar les categories disponibles per a l'usuari. Es mostrarà amb una llista on estaran els botons per editar i eliminar les categories. Això inclou una validació per a la seva modificació.

- btList: Ens serveix per refrescar el llistat en cas de ser necessari.
- btInput: Ens retornarà a la pàgina d'afegir categories.
- categoryList:Ens mostra la llista de categories.

Explicació JS Categories

CategoriesNoticies

```
function main(){
  // Se seleccionan los elementos del HTML necesarios
  const sendCategoryButton = document.querySelector("#btSend")
  const listCategoryButton = document.querySelector("#btList")
  const nameCategoryText = document.querySelector("#category")
  const categoryList = document.querySelector("#categoryList")
  const messageDebugg = document.querySelector("#message")

  // Se muestran las categorías guardadas al cargar la página
  displayCategory(categoryList)

  // Se añade un evento al botón para guardar una nueva categoría
  sendCategoryButton.addEventListener("click", () => {   // Se obtiene el valor del input
    addCategory(nameCategoryText, messageDebugg, categoryList)

    })

    listCategoryButton.addEventListener("click", () => {
      window.location.href = "./CategoriesLlistat.html"
    })
}
```

Ací estem seleccionant elements del HTML i els guardem en constants per poder-los manipular després amb JavaScript:

- sendCategoryButton: el botó per a enviar o guardar una nova categoria.
- listCategoryButton: el botó per a mostrar el llistat de categories.
- nameCategoryText: el camp on l'usuari escriu el nom de la categoria.
- categoryList: l'element on es mostraran les categories guardades.
- messageDebugg: un element per a mostrar missatges de confirmació o errors.

Aquí afegim un "event listener" al botó de guardar categoria. Quan l'usuari faça clic:
Es llig el nom que ha escrit l'usuari (nameCategoryText).
Es crida la funció addCategory que guarda la nova categoria i actualitza la llista (categoryList) i els missatges (messageDebugg).

I ací, quan l'usuari fa clic al botó de "llistat de categories", la pàgina es redirigeixa l'altre HTML

```
function addCategory(input, message, container){
  const nameCategory = input.value.trim()

  // Validació del nom
  if(!validationName(nameCategory, message)){
    return false;
  }

        if (nameCategory) {
          // Si ya existen categorías en localStorage, se recuperan, sino se inicializa un array vacío
          const existingCategories = localStorage.getItem("category") ? JSON.parse(localStorage.getItem("category")) : [];
          // Se añade la nueva categoría al array
          existingCategories.push(nameCategory);
          // Se guarda el array actualizado en localStorage
          localStorage.setItem("category", JSON.stringify(existingCategories));
          message.innerHTML = "Categoria guardada correctament."  // Se muestra un mensaje confirmando el guardado
          input.value = ""
        } else {
            displayCategory(container)
            message.innerHTML = "No s'ha introduït cap categoria."  // Si no se ha escrito nada, se muestra un mensaje
            }
}
```

Aquesta és la funció addCategory. Rep tres paràmetres:

- input: l'element on l'usuari escriu el nom de la categoria.
- message: l'element on es mostraran missatges d'informació o errors.
- container: l'element on es mostrarà la llista de categories.

Es comprova que el nom siga vàlid amb la funció validationName. Si no ho és, la funció s'atura (return false) i no guarda res.

Despres de la validacio:

- Mirem si ja hi ha categories guardades en localStorage.
- Si n'hi ha, les convertim de text (JSON) a array amb JSON.parse.
- Si no hi ha categories, inicialitzem un array buit [].
- Afexim la categoria a l'array
- O guardem en el localStorage
- Es mostrar un misastje en cas de guardarse correctament o no

```
//Funcio per a la validacio del nom
function validationName(categoria, messageElement) {
const element = document.getElementById("category");

if (!element.checkValidity()) {
  if (element.validity.valueMissing) {
    error(messageElement, "Has d'introduir una categoria");
  } else if (element.validity.patternMismatch) {
    error(messageElement, "La categoria ha de tindre entre 2 i 20 caracters i començar amb majúscula.");
  } else {
    error(messageElement, "Categoria no vàlida");
  }
  return false;
}
return true;
}
```

Aquesta és la funció validationName. Rep dos paràmetres:

- categoria: el nom de la categoria que volem validar.
- messageElement: l'element HTML on es mostraran els missatges d'error.

CheckValidity

- checkValidity() comprova si l'input compleix les regles del HTML (per exemple, required, pattern).
- Si no és vàlid, entrem dins del if.
- Si l'usuari no ha escrit res (valueMissing), cridem la funció error per a mostrar un missatge: "Has d'introduir una categoria".
- Si el valor escrit no coincideix amb el patró definit al HTML (patternMismatch), mostrem un missatge indicant la regla: ha de tenir entre 2 i 20 caràcters i començar amb majúscula.
- Si hi ha qualsevol altre error de validació, mostrem un missatge genèric: "Categoria no vàlida".
- Retornem false si la validació falla i true si tot és correcte.

CategoriaLlista

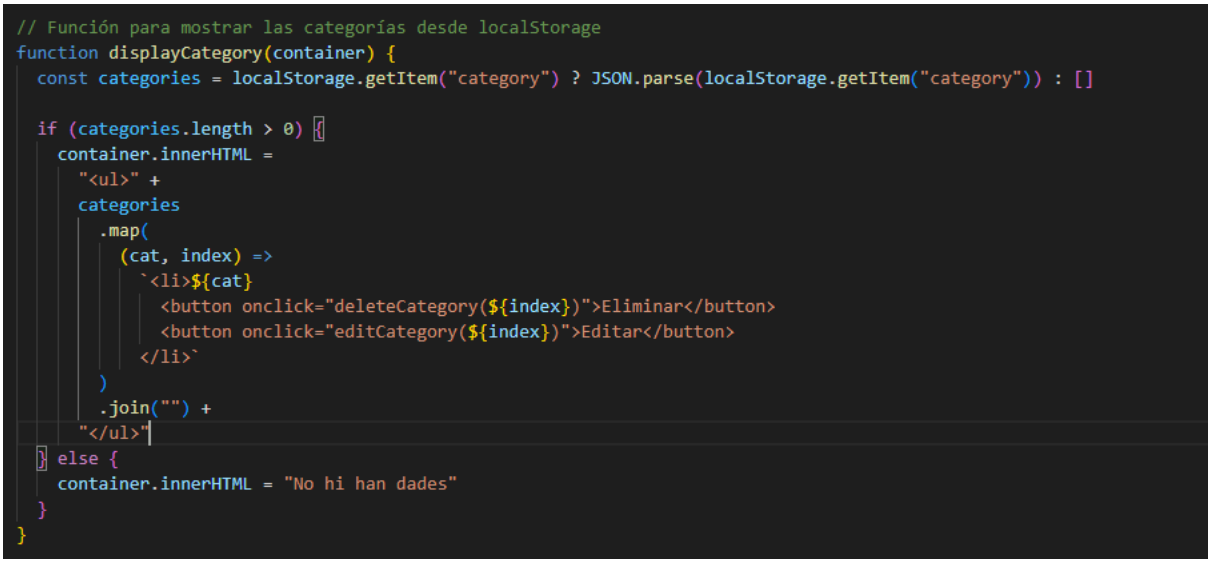
```
function main() {
  const categoryList = document.querySelector("#categoryList")
  const addCategoryButton = document.querySelector("#btInput")
  const listCategoryButton = document.querySelector("#btList")

  // Mostrar categorías guardadas
  displayCategory(categoryList)

  // Botón para volver a añadir categoría
  addCategoryButton.addEventListener("click", () => {
    window.location.href = "./CategoriesNoticies.html"
  })

  // Botón para refrescar lista
  listCategoryButton.addEventListener("click", () => {
    displayCategory(categoryList)
  })
}
```

Seleccionem els elements del HTML que necessitarem:

- categoryList: l'element on es mostrarà la llista de categories.
- addCategoryButton: el botó per anar a la pàgina on es poden afegir categories noves.
- listCategoryButton: el botó per actualitzar/refrescar la llista de categories.

displayCategory
Mostrem les categories que ja estan guardades, cridant la funció displayCategory i passant-li l'element on es mostraran.

Redireccio per EventListener en el button
Quan l'usuari fa clic al botó d'afegir categoria, es redirigeix a la pàgina CategoriesNoticies.html on probablement es poden afegir categories noves.



```
// Función para mostrar las categorías desde localStorage
function displayCategory(container) {
  const categories = localStorage.getItem("category") ? JSON.parse(localStorage.getItem("category")) : []

  if (categories.length > 0) {
    container.innerHTML =
      "<ul>" +
      categories
        .map(
          (cat, index) =>
            `<li>${cat}
              <button onclick="deleteCategory(${index})">Eliminar</button>
              <button onclick="editCategory(${index})">Editar</button>
            </li>`
        )
        .join("") +
      "</ul>"
  } else {
    container.innerHTML = "No hi han dades"
  }
}
```

Aquesta és la funció displayCategory que serveix per a mostrar les categories guardades en el localStorage. Rep un paràmetre:

- container: l'element HTML on es mostrarà la llista de categories.
- Mirem si hi ha categories guardades al localStorage.
- Si n'hi ha, les convertim de text (JSON) a un array amb JSON.parse.
- Si no n'hi ha, inicialitzem un array buit []
- Si l'array té categories, procedim a mostrar-les.
- Creem una llista HTML (<ul>).
- Per cada categoria (cat) fem un <li> amb:
    - El nom de la categoria.
    - Un botó per eliminar-la (deleteCategory(index)), passant-li l'índex de la categoria.
    - Un botó per editar-la (editCategory(index)), també amb l'índex.
- map() genera un array de línies HTML i join("") les uneix en un sol text.
- Finalment, tot s'assigna a container.innerHTML perquè aparega a la pàgina
- Si no hi han guardades es mostrara un missatge

```
// Función para eliminar categorías
function deleteCategory(index) {
  const categories = JSON.parse(localStorage.getItem("category")) || []
  const confirmDelete = confirm(`¿Segur que vols eliminar la categoria ${categories[index]}?`)
  if (confirmDelete) {
    categories.splice(index, 1)
    localStorage.setItem("category", JSON.stringify(categories))
    displayCategory(document.querySelector("#categoryList"))
  }
}
```

Aquesta és la funció deleteCategory. Rep un paràmetre:

- index: l'índex de la categoria que volem eliminar dins de l'array de categories.
- Recuperem les categories guardades al localStorage.
- Si no hi ha categories, inicialitzem un array buit [].
- Mostrem un missatge de confirmació a l'usuari amb confirm().
- Li preguntem si està segur que vol eliminar la categoria concreta (categories[index]).
- El resultat (true o false) s'emmagatzema a confirmDelete.
- Si es TRue selimina l'usuari
- Es torna a cridar a la catgeory per actualitzar la llista

```
function editCategory(index) {
  const categories = JSON.parse(localStorage.getItem("category")) || [];
  const oldName = categories[index];
  const newName = prompt("Escriu el nou nom per a la categoria:", oldName);

  if (newName && newName.trim() !== "") {
    if (!validateCategoryName(newName.trim())) {
      alert("La categoria ha de tenir entre 2 i 20 lletres, començar amb majúscula i només lletres.");
      return;
    }
    categories[index] = newName.trim();
    localStorage.setItem("category", JSON.stringify(categories));
    displayCategory(document.querySelector("#categoryList")); //Actualització de la llista
  }
}
```

Aquesta és la funció editCategory. Rep un paràmetre:

- index: l'índex de la categoria que volem modificar dins de l'array de categories.
- Recuperem les cartegories guardades
- Es guarda el nom de les categories per editar
- Es mostra un prompt per a modificar el nom
- Es valida el nom
- Si la validación es correcta es guardar
- Es torna a refrescar la llista de catgeoria

```
// Validacio dels noms cuan s'escriuen
function validateCategoryName(name) {
  const pattern = /^[A-Z][a-z]{1,19}$/; // 2 a 20 lletres, comença amb majúscula
  return pattern.test(name);
}
```

Aquesta és la funció validateCategoryName. Rep un paràmetre:

- name: el nom de la categoria que volem validar.
- Definim un patró (regex) per a comprovar que el nom siga correcte:
  - ^[A-Z] → el primer caràcter ha de ser una majúscula.
  - [a-z]{1,19} → els següents 1 a 19 caràcters han de ser lletres minúscules.
  - $ → final del text, assegurant que només hi haja aquests caràcters.
- En total, el nom ha de tindre entre 2 i 20 lletres i començar amb majúscula.
- pattern.test(name) comprova si el nom coincideix amb el patró.
- Retorna true si és vàlid i false si no ho és.

Explicació HTML Usuari

Usuari

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Usuaris formulary</title>
    <script type="text/javascript" src="usuaris.js"></script>
</head>
<body>
    <h1>Usuaris</h1>
    <form>
        <label>Nom</label>
        <br><br>
        <input type="text" placeholder="Nom del usuari" id="nom">
        <br><br>
        <label>Nom usuari</label>
        <br><br>
        <input type="text" placeholder="Nom del usuari" id="nomUsuari">
        <br><br>
        <label>Contrasenya</label>
        <br><br>
        <input type="password" placeholder="Contrasenya" id="contrasenya">
        <br><br>
        <label>Email</label>
        <br><br>
        <input type="email" placeholder="Email" id="email">
        <br><br>
        <button type="button" id="btSend">Enviar</button>
        <button type="button" id="btEdit">Editar Usuaris</button>
    </form>

    <p id="message"></p>
    <div id="nomUsuariList"></div>
</body>
</html
```

Es mostrarà a l'usuari un formulari bàsic on podrà introduir nom, nickname, contrasenya i email. També hi haurà uns botons per enviar i editar l'usuari.

- btSend: Enviar les dades al nostre
- btEdit:En portara a la pagina per editar usuari

editarUsuari

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Editar Usuaris</title>
    <script type="text/javascript" src="editarUsuaris.js"></script>
</head>
<body>
    <h1>Editor d'usuaris</h1>
    <button type="button" id="btCreate">Crear usuaris</button>
    <br><br>
    <table border="1" id="usersTable">
    <thead>
        <tr>
            <th>Nom real</th>
            <th>Nom usuari</th>
            <th>Email</th>
            <th>Contrasenya</th>
            <th>Accions</th>
        </tr>
    </thead>
    <tbody>

    </tbody>
</table>
</body>
</html>
```

Mostrarà una taula generada per JS on es veuran els usuaris creats amb les seues dades corresponents.

Explicació JS Usuari

Usuari

```
function main(){
// Se seleccionan los elementos del HTML necesarios
const sendUsuariButton = document.querySelector("#btSend")
const editUseriButton = document.querySelector("#btEdit")
const nomText = document.querySelector("#nom")
const nameUsuariText = document.querySelector("#nomUsuari")
const nomUsuariList = document.querySelector("#nomUsuariList")
const passwordUsuariText = document.querySelector("#contrasenya")
const emailUsuariText = document.querySelector("#email")
const messageDebugg = document.querySelector("#message")

// Se muestran los usuario guardadas al cargar la página
displayUsuari()

// Se añade un evento al botón para guardar un nuevo usuario
sendUsuariButton.addEventListener("click", () => {   // Se obtiene el valor del input
        const nom = nomText.value.trim()
        const nameUsuari = nameUsuariText.value.trim()
        const passwordUsuari = passwordUsuariText.value.trim()
        const emailUsuari = emailUsuariText.value.trim()
        // Validacio dels camps
        if (!validateNombreReal(nom)) return;
        if (!validateNickname(nameUsuari)) return;
        if (!validatePassword(passwordUsuari)) return;
        if (!validateEmail(emailUsuari)) return;

        // Si ya existen categorías en localStorage, se recuperan, sino se inicializa un array vacío
        const existingNom = localStorage.getItem("nom") ? JSON.parse(localStorage.getItem("nom")) : [];
        const existingNomUsuari = localStorage.getItem("nomUsuari") ? JSON.parse(localStorage.getItem("nomUsuari")) : [];
        const existingcontrasenya = localStorage.getItem("contrasenya") ? JSON.parse(localStorage.getItem("contrasenya")) : [];
        const existingEmailUsuari = localStorage.getItem("email") ? JSON.parse(localStorage.getItem("email")) : [];

        // Se añaden los nuevos datos al array
        existingNom.push(nom)
        existingNomUsuari.push(nameUsuari);
        existingcontrasenya.push(passwordUsuari);
        existingEmailUsuari.push(emailUsuari);
        // Se guarda el array actualizado en localStorage
        localStorage.setItem("nom", JSON.stringify(existingNom))
        localStorage.setItem("nomUsuari", JSON.stringify(existingNomUsuari));
        localStorage.setItem("contrasenya", JSON.stringify(existingcontrasenya));
        localStorage.setItem("email", JSON.stringify(existingEmailUsuari));

        //Misssatge de debug
        messageDebugg.innerHTML = "Usuari guardada correctament."  // Se muestra un mensaje confirmando el guardado

        nomText.value = "";
        nameUsuariText.value = "";
        passwordUsuariText.value = "";
        emailUsuariText.value = "";
})
```

Seleccionem els elements HTML necessaris:

- sendUsuariButton: botó per guardar un nou usuari.
- editUseriButton: botó per editar usuaris (redirigeix a altra pàgina).
- nomText: input del nom real de l'usuari.
- nameUsuariText: input del nom d'usuari (nickname).
- nomUsuariList: element on es poden mostrar els usuaris (llista).
- passwordUsuariText: input de la contrasenya.
- emailUsuariText: input del correu electrònic.
- messageDebugg: element per mostrar missatges d'informació.

- Agafem els valors escrits per l'usuari i eliminem espais al principi i final amb trim().
- Validem cada camp amb funcions específiques. Si alguna no és vàlida, aturem la funció (return).
- Recuperem les dades existents del localStorage.
- Si no existeixen, inicialitzem arrays buits.
- Afegim les noves dades als arrays corresponents.
- Guardem els arrays actualitzats al localStorage.
- Mostrem un missatge indicant que l'usuari s'ha guardat correctament.
- Netejem els camps del formulari per a poder introduir un altre usuari.
- Quan es fa clic al botó d'editar usuaris, redirigeix a la pàgina editarUsuaris.html.

```
function displayUsuari() {
  // Se obtienen los usuarios de localStorage (si no existen, se usa un array vacío)
  const nomFromLocalStorage = localStorage.getItem("nom") ? JSON.parse(localStorage.getItem("nom")) : []
  const nameFromLocalStorage = localStorage.getItem("nomUsuari") ? JSON.parse(localStorage.getItem("nomUsuari")) : []
  const passwordFromLocalStorage = localStorage.getItem("contrasenya") ? JSON.parse(localStorage.getItem("contrasenya")) : []
  const emailFromLocalStorage = localStorage.getItem("email") ? JSON.parse(localStorage.getItem("email")) : []

  console.log("hola " + nameFromLocalStorage)  // Se imprime en consola para depuración

  if (nameFromLocalStorage.length > 0) {
    // Si existen categorías,
    nomUsuariList.innerHTML = nameFromLocalStorage
  } else {
    // Si no hay categorías, se muestra un mensaje en el contenedor
    nomUsuariList.innerHTML = "No hi han dades"
  }
}
```

(const nameFromLocalStorage: any)

Aquesta és la funció displayUsuari. S'encarrega de mostrar els usuaris guardats al localStorage en la pàgina.

- Recuperem les dades dels usuaris guardades al localStorage.
- Si no existeixen, inicialitzem arrays buits ([]):
  - nomFromLocalStorage: noms reals.
  - nameFromLocalStorage: noms d'usuari (nicknames).
  - passwordFromLocalStorage: contrasenyes.
  - emailFromLocalStorage: correus electrònics.
- Mostrem a la consola el contingut de nameFromLocalStorageper a depurar i comprovar que es recuperen els usuaris correctament.
- Si hi ha usuaris guardats (length > 0), els mostrem al contenidor nomUsuariList.
- Si no hi ha dades, mostrem un missatge indicant-ho: "No hi han dades".

```javascript
// Validacio de la informacio
function validateNombreReal(name) {
    const pattern = /^[A-Z][a-zA-Z]{1,19}$/;
    if (!pattern.test(name.trim())) {
        alert("Nom real: 2-20 lletres, comença amb majúscula, sense números.");
        return false;
    }
    return true;
}

function validateNickname(name) {
    const pattern = /^[A-Z][a-zA-Z0-9]{1,19}$/;
    if (!pattern.test(name.trim())) {
        alert("Nom usuari: 2-20 caràcters, comença amb majúscula, números permesos.");
        return false;
    }
    return true;
}

function validatePassword(password) {
    const pattern = /^(?=.*[A-Z])(?=.*[\W_]).+$/;
    if (!pattern.test(password)) {
        alert("Contrasenya: ha de contenir almenys una majúscula i un símbol.");
        return false;
    }
    return true;
}

function validateEmail(email) {
    const pattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!pattern.test(email.trim())) {
        alert("Email: format invàlid, ha de contenir '@' i domini.");
        return false;
    }
    return true;
}
```

Validació Nom

- Aquesta funció valida el nom real d'un usuari.
- Patró:
    - ^[A-Z] → comença amb majúscula.
    - [a-zA-Z]{1,19} → de 1 a 19 lletres més (majúscules o minúscules).
- En total, el nom ha de tindre entre 2 i 20 lletres, sense números ni símbols.
- Retorna true si és vàlid, false si no, mostrant un alert amb l'error.

Validació NickName

- Aquesta funció valida el nickname o nom d'usuari.
- Patró:
    - ^[A-Z] → comença amb majúscula.
    - [a-zA-Z0-9]{1,19} → de 1 a 19 caràcters addicionals, poden ser lletres o números.
- Total: entre 2 i 20 caràcters, números permesos, sense símbols.
- Retorna true si és vàlid, false si no.

Validació Password

- Aquesta funció valida la contrasenya.
- Patró:
    - (?=.*[A-Z]) → almenys una lletra majúscula.
    - (?=.*[\W_]) → almenys un símbol o caràcter especial (no lletra ni número).
- Retorna true si compleix ambdues condicions, false si no, mostrant un alert.

Validació Email

Aquesta funció valida el correu electrònic.

- Patró:
    - Ha de tenir alguna cosa abans de @, després un @, i un domini amb punt (ex: exemple@domini.com).
    - No permet espais.
- Retorna true si és vàlid, false si no.

editarUsuari

```
function getUsuarios() {
    return {
        nom: JSON.parse(localStorage.getItem("nom") || "[]"),
        nomUsuari: JSON.parse(localStorage.getItem("nomUsuari") || "[]"),
        email: JSON.parse(localStorage.getItem("email") || "[]"),
        contrasenya: JSON.parse(localStorage.getItem("contrasenya") || "[]")
    };
}
```

Aquesta és la funció getUsuarios. No rep cap paràmetre i serveix per recuperar totes les dades dels usuaris del localStorage.

- Retorna un objecte amb quatre propietats:
    - nom: array amb els noms reals dels usuaris.
    - nomUsuari: array amb els noms d'usuari (nicknames).
    - email: array amb els correus electrònics.
    - contrasenya: array amb les contrasenyes.
- Per a cada propietat:
    - Recuperem les dades del localStorage amb localStorage.getItem().
    - Si no existeixen (null), utilitzem un array buit "[]".
    - Convertim les dades de text a array amb JSON.parse().

```
function mostrarUsuarios() {
    const users = getUsuarios();
    tbody.innerHTML = "";

    if (users.nomUsuari.length === 0) {
        tbody.innerHTML = `<tr><td colspan="5">No hi ha usuaris</td></tr>`;
        return;
    }

    users.nomUsuari.forEach((nombre, i) => {
        tbody.innerHTML += `
            <tr>
                <td>${users.nom[i]}</td>
                <td>${nombre}</td>
                <td>${users.email[i]}</td>
                <td>${"*".repeat(users.contrasenya[i].length)}</td>
                <td>
                    <button onclick="editarUsuario(${i})">Editar</button>
                    <button onclick="eliminarUsuario(${i})">Eliminar</button>
                </td>
            </tr>
        `;
    });
}
```

Aquesta funció serveix per mostrar la llista d'usuaris en una taula HTML (tbody).

- Recuperem totes les dades dels usuaris cridant la funció getUsuarios().
- Netejem el contingut de la taula abans de mostrar les dades noves.
- Si no hi ha usuaris (length === 0), mostrem un missatge en una fila de la taula: "No hi ha usuaris".
- colspan="5" fa que el missatge ocupe totes les columnes de la taula.
- Retornem (return) per a que la funció no continue.
- Recorrem cada usuari amb forEach.
- Per cada usuari:
    - Mostrem el nom real (users.nom[i]).
    - Mostrem el nickname (nombre).
    - Mostrem el correu electrònic (users.email[i]).
    - Mostrem la contrasenya encriptada visualment amb asteriscs ("*".repeat(users.contrasenya[i].length)).
    - Afegim botons per editar (editarUsuario(i)) o eliminar (eliminarUsuario(i)) l'usuari corresponent.
- Cada iteració afegeix una nova fila (<tr>) al tbody.

```
    window.editarUsuario = function(i) {
    const users = getUsuarios();

    const nuevoNom = prompt("Nou nom real:", users.nom[i]);
    if (nuevoNom === null || !validateNombreReal(nuevoNom.trim())) return;

    const nuevoNombre = prompt("Nou nom usuari:", users.nomUsuari[i]);
    if (nuevoNombre === null || !validateNickname(nuevoNombre.trim())) return;

    const nuevoEmail = prompt("Nou email:", users.email[i]);
    if (nuevoEmail === null || !validateEmail(nuevoEmail.trim())) return;

    const nuevaContrasenya = prompt("Nova contrasenya:", users.contrasenya[i]);
    if (nuevaContrasenya === null || !validatePassword(nuevaContrasenya)) return;

    // Si todas las validaciones pasan, se guardan los datos
    users.nom[i] = nuevoNom.trim();
    users.nomUsuari[i] = nuevoNombre.trim();
    users.email[i] = nuevoEmail.trim();
    users.contrasenya[i] = nuevaContrasenya.trim();

    localStorage.setItem("nom", JSON.stringify(users.nom));
    localStorage.setItem("nomUsuari", JSON.stringify(users.nomUsuari));
    localStorage.setItem("email", JSON.stringify(users.email));
    localStorage.setItem("contrasenya", JSON.stringify(users.contrasenya));

    alert("Usuari actualitzat correctament!");
    mostrarUsuarios();
};
```

Aquesta funcio esta fet per poder cridar-se desde el button del HTML i poder

- Recuperem totes les dades dels usuaris del localStorage en un objecte amb arrays.
- Demanem un nou nom real amb prompt, mostrant el nom antic per defecte.
- Si l'usuari prem "Cancel" (null) o el nom no passa la validació, aturem la funció.
- Demanem un nou nickname amb prompt.
- Igual que abans, aturem si és cancel·lat o no vàlid.
- Demanem un nou email.
- Si és cancel·lat o no vàlid, aturem.
- Demanem una nova contrasenya.
- Aturem si no és vàlida o es cancel·la.
- Si totes les validacions passen, actualitzem les dades de l'usuari en els arrays corresponents.
- Guardem els arrays actualitzats al localStorage.
- Mostrem un missatge indicant que l'usuari s'ha actualitzat correctament.
- Tornem a cridar mostrarUsuarios() per actualitzar la taula amb les dades noves.

```
    window.eliminarUsuario = function(i) {
        const users = getUsuarios();

        if (confirm(`Segur que vols eliminar l'usuari "${users.nomUsuari[i]}"?`)) {
            users.nom.splice(i, 1);
            users.nomUsuari.splice(i, 1);
            users.email.splice(i, 1);
            users.contrasenya.splice(i, 1);

            localStorage.setItem("nom", JSON.stringify(users.nom));
            localStorage.setItem("nomUsuari", JSON.stringify(users.nomUsuari));
            localStorage.setItem("email", JSON.stringify(users.email));
            localStorage.setItem("contrasenya", JSON.stringify(users.contrasenya));

            alert("Usuari eliminat correctament!");
            mostrarUsuarios();
        }
    };
});
```

Aquesta funcio esta fet per poder cridar-se desde el button del HTML i poder

- Recuperem totes les dades dels usuaris del localStorage en un objecte amb arrays.
- Mostrem una finestra de confirmació (confirm) preguntant si està segur de voler eliminar aquest usuari.
- Si l'usuari prem "Cancel", la funció s'atura automàticament.
- Si confirma, eliminem l'usuari de tots els arrays (nom, nomUsuari, email, contrasenya) amb splice(i, 1).
- Guardem els arrays actualitzats al localStorage, reemplaçant les dades anteriors.
- Mostrem un missatge indicant que l'usuari s'ha eliminat correctament.
- Tornem a cridar mostrarUsuarios() per actualitzar la taula amb les dades noves.

```
//Funcions per a la validacio dels camps
function validateNombreReal(name) {
    const pattern = /^[A-Z][a-zA-Z]{1,19}$/;
    if (!pattern.test(name.trim())) {
        alert("Nom real: 2-20 lletres, comença amb majúscula, sense números.");
        return false;
    }
    return true;
}

function validateNickname(name) {
    const pattern = /^[A-Z][a-zA-Z0-9]{1,19}$/;
    if (!pattern.test(name.trim())) {
        alert("Nom usuari: 2-20 caràcters, comença amb majúscula, números permesos.");
        return false;
    }
    return true;
}

function validatePassword(password) {
    const pattern = /^(?=.*[A-Z])(?=.*[\W_]).+$/;
    if (!pattern.test(password)) {
        alert("Contrasenya: ha de contenir almenys una majúscula i un símbol.");
        return false;
    }
    return true;
}

function validateEmail(email) {
    const pattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!pattern.test(email.trim())) {
        alert("Email: format invàlid, ha de contenir '@' i domini.");
        return false;
    }
    return true;
}
```

Validació Nom

- Patró (regex):
  - ^[A-Z] → el primer caràcter ha de ser majúscula.
  - [a-zA-Z]{1,19} → de 1 a 19 caràcters més (majúscules o minúscules).
- En total, el nom ha de tindre entre 2 i 20 lletres, sense números ni símbols.
- Retorna true si és vàlid, false si no i mostra un missatge d'error.

Validació Nickname

- Patró:
  - Comença amb majúscula (^[A-Z]).
  - Seguit de 1 a 19 caràcters que poden ser lletres o números ([a-zA-Z0-9]{1,19}).
- Total: entre 2 i 20 caràcters, números permesos, sense símbols.

Validació Password

- Patró:
  - (?=.*[A-Z]) → almenys una lletra majúscula.
  - (?=.*[\W_]) → almenys un símbol o caràcter especial (no lletra ni número).
- Retorna true si compleix ambdues condicions, false si no.

Validació Email

- Patró:
  - Ha de tindre algun text abans de @, després un @ i un domini amb punt (ex: exemple@domini.com).
  - No permet espais.
- Retorna true si és vàlid, false si no.