PROYECTO SALESFORCE

Alejandro ESTEBAN GIL - David MOLINOS GRACIA

Crear componentes web Lightning

PRIMER MÓDULO LWC

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata" fqn="helloWorld">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

helloWorld.html | helloWorld.js | helloWorld.js-meta.xml

force-app > main > default > lwc > helloWorld > helloWorld.html > template > lightning-card > div.slds-m-around_medium

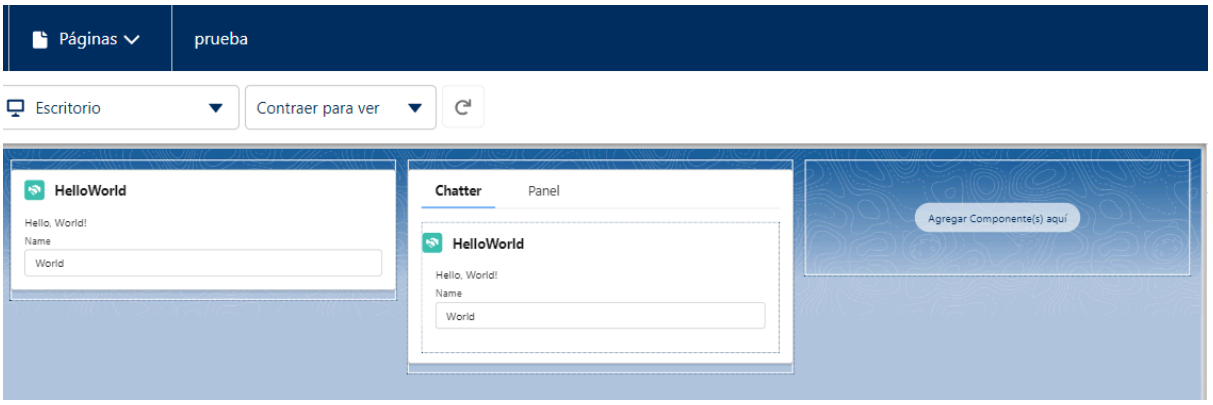
```html
<template>
    <lightning-card title="HelloWorld" icon-name="custom:custom14">
        <div class="slds-m-around_medium">
            <p>Hello, {greeting}!</p>
            <lightning-input label="Name" value={greeting} onchange={changeHandler}></lightning-input>
        </div>
    </lightning-card>
</template>
```

helloWorld.html | helloWorld.js | helloWorld.js-meta.xml

force-app > main > default > lwc > helloWorld > helloWorld.js > HelloWorld > changeHandler

```boilerplate
import { LightningElement } from 'lwc';

export default class HelloWorld extends LightningElement {
    greeting = 'World';
    changeHandler(event) {
        this.greeting = event.target.value;
    }
}
```



Otra forma de crear un LWC

```
C:\Users\David\Desktop\DAVID\00.-FP\04.-SGE\SALESFORCE\PROYECTO\Proyecto-Salesforce\proyecto>sf lightning generate component -n myF
irstWebComponent -d force-app/main/default/lwc --type lwc
```

-n : nombre del LWC

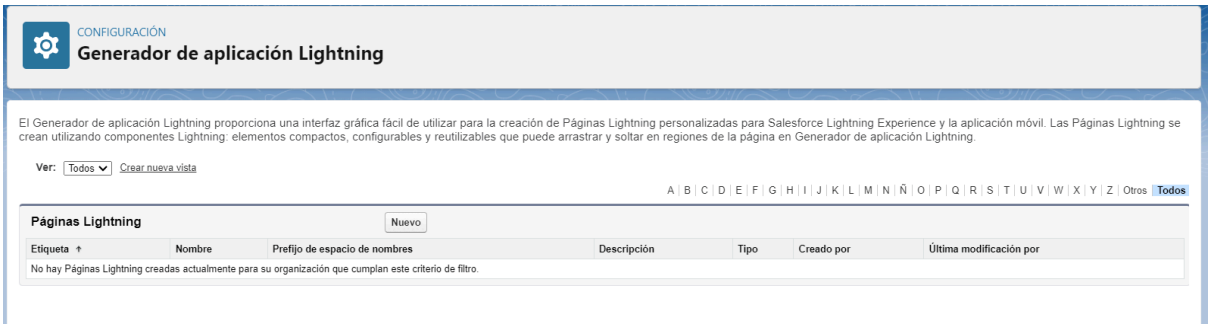
-d : la ruta donde se creará

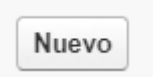
–type : indica que será un lwc

```
C:\Users\David\Desktop\DAVID\00.-FP\04.-SGE\SALESFORCE\PROYECTO\Proyecto-Salesforce\proyecto>sf lightning generate component -n myF
irstWebComponent -d force-app/main/default/lwc --type lwc
 »   Warning: @salesforce/cli update available from 2.20.6 to 2.30.8.
target dir = C:\Users\David\Desktop\DAVID\00.-FP\04.-SGE\SALESFORCE\PROYECTO\Proyecto-Salesforce\proyecto\force-app\main\default\lw
c
   create force-app\main\default\lwc\myFirstWebComponent\myFirstWebComponent.js
   create force-app\main\default\lwc\myFirstWebComponent\myFirstWebComponent.html
   create force-app\main\default\lwc\myFirstWebComponent\__tests__\myFirstWebComponent.test.js
   create force-app\main\default\lwc\myFirstWebComponent\myFirstWebComponent.js-meta.xml
```

Para incorporar los LWC se puede seguir:

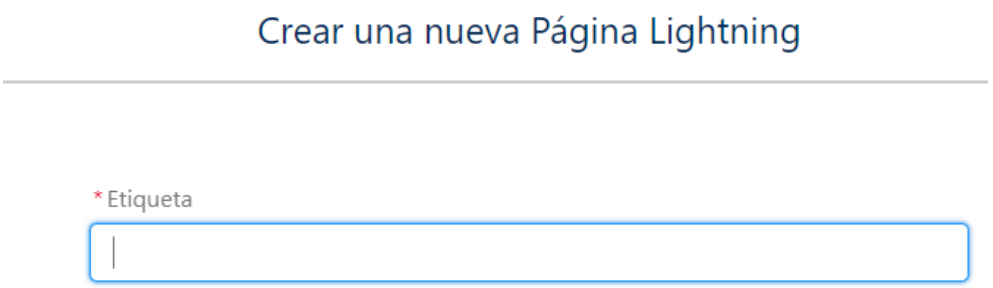
Primero vamos al Generador de aplicación Lightning



Seleccionamos 

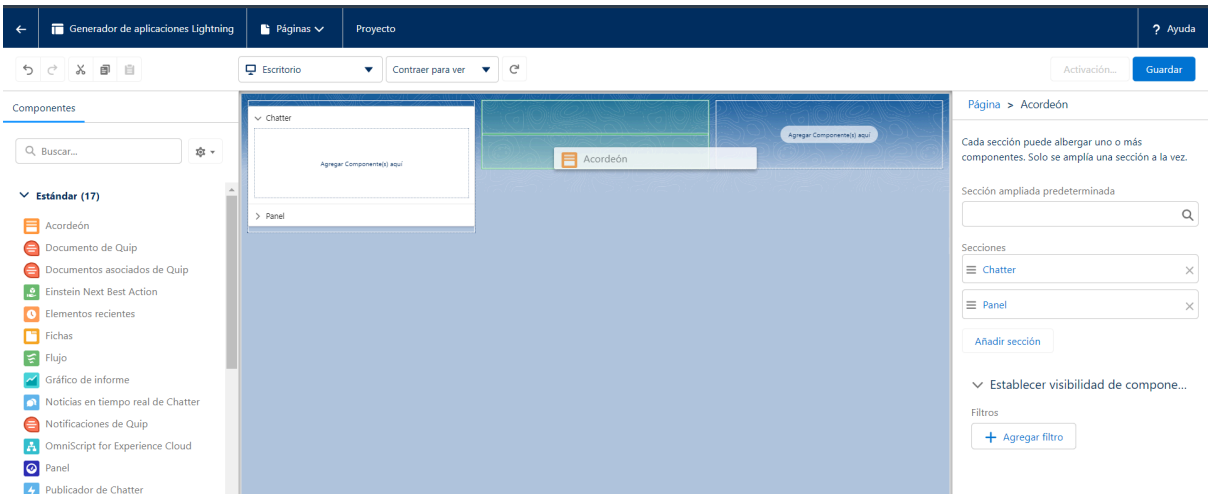
Elegimos que tipo de página queremos:

Escribimos el nombre de la página:

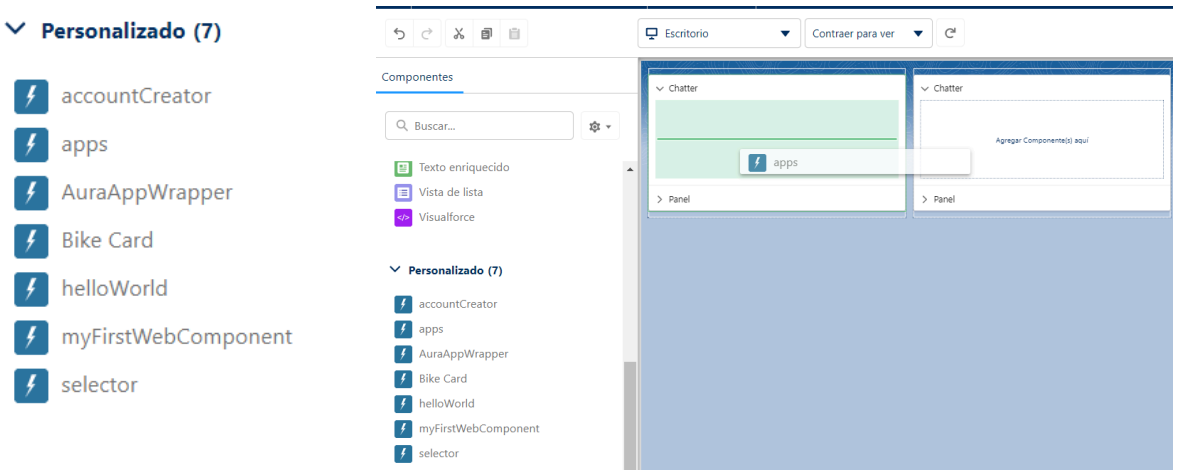


Elegimos la estructura por defecto que tendrá la página:

Creamos la página arrastrando los componentes:



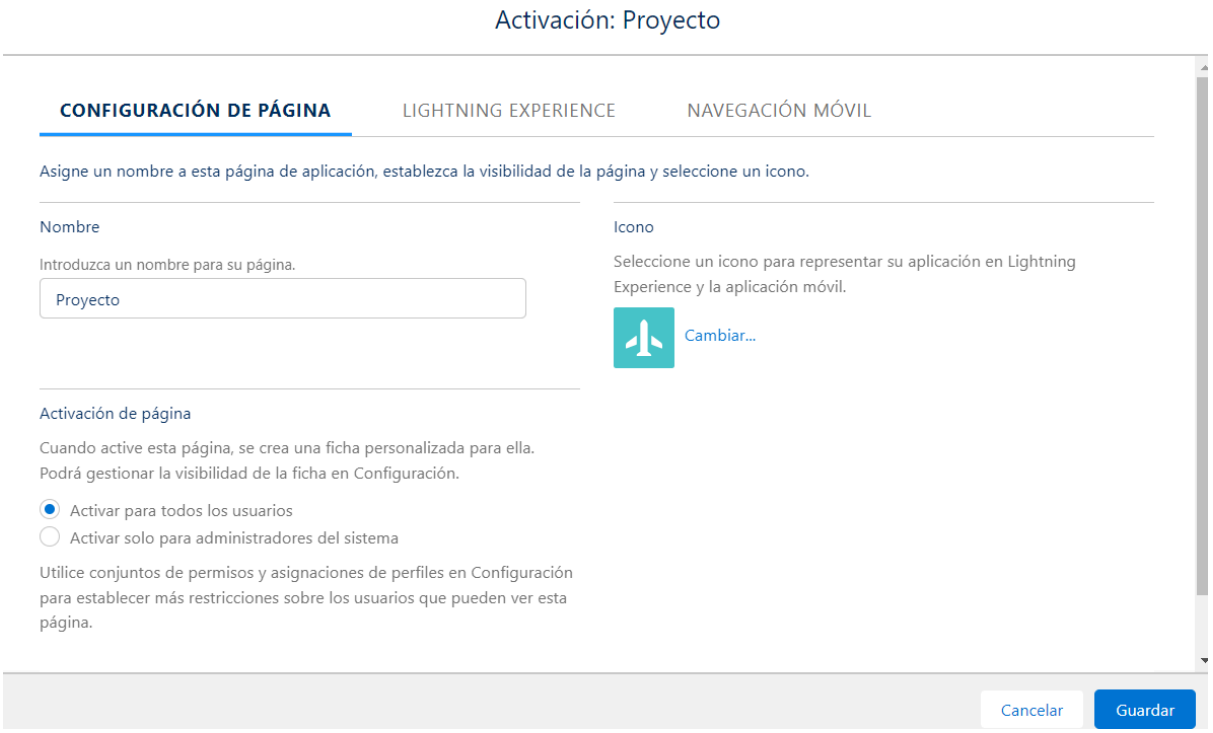
Podemos añadir componentes estándar o personalizados.



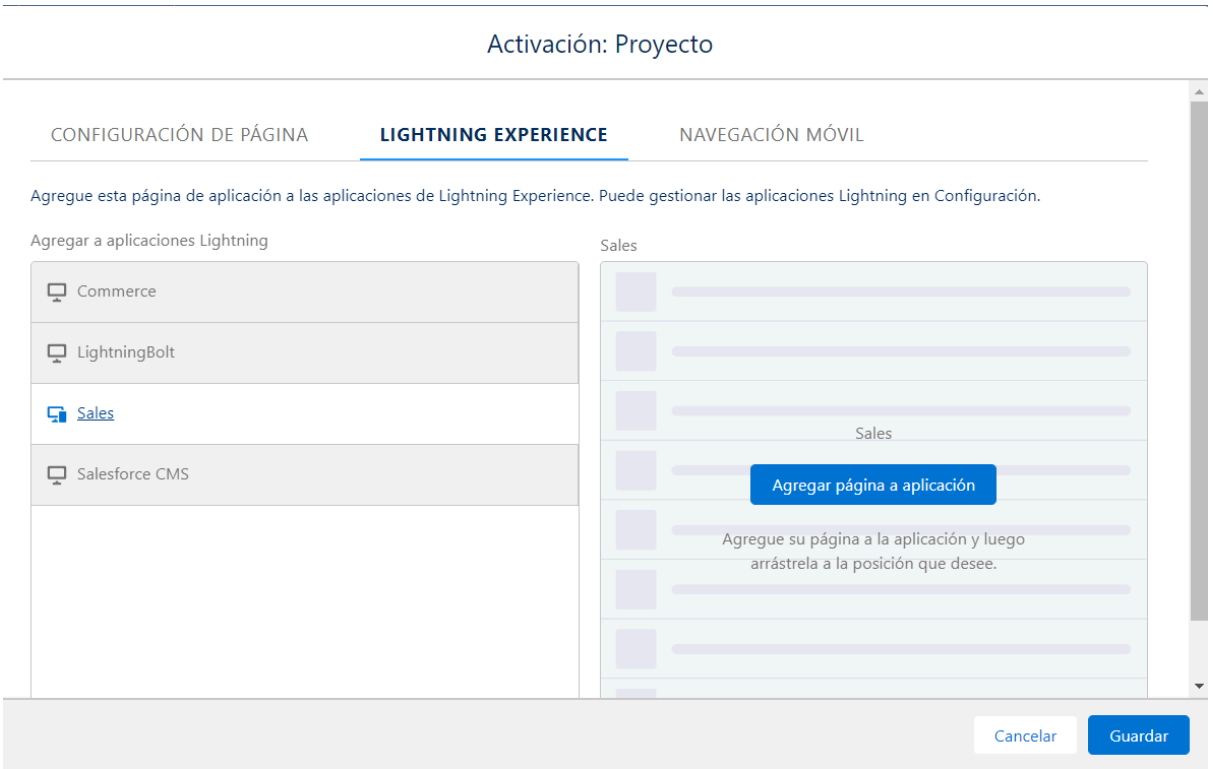
Pulsamos

Y después 

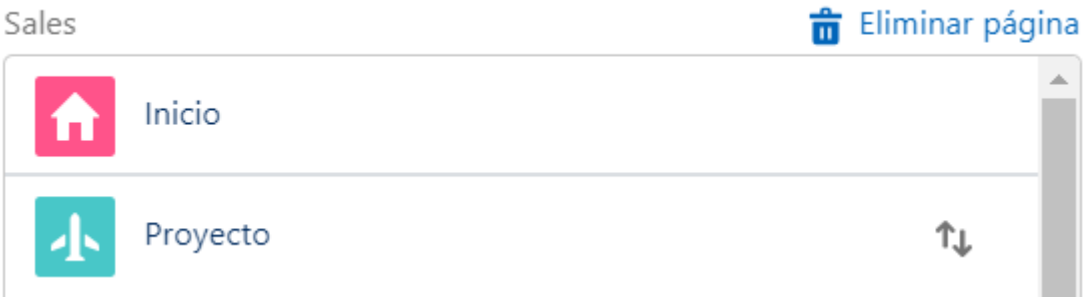
En activación nos sale la siguiente ventana:

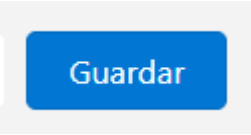


Donde podremos cambiar el nombre y el icono, Cambiamos a la pestaña:

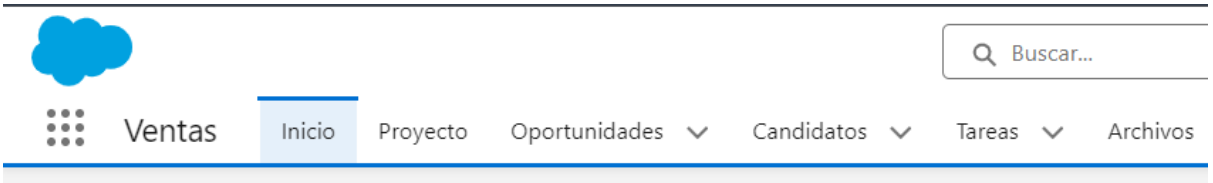


En la que elegiremos la aplicación creada por defecto que queramos y añadiremos nuestra página.



Salimos pulsando 

En Ventas(Sales) aparecerá nuestra página:



SEGUNDO MÓDULO LWC

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

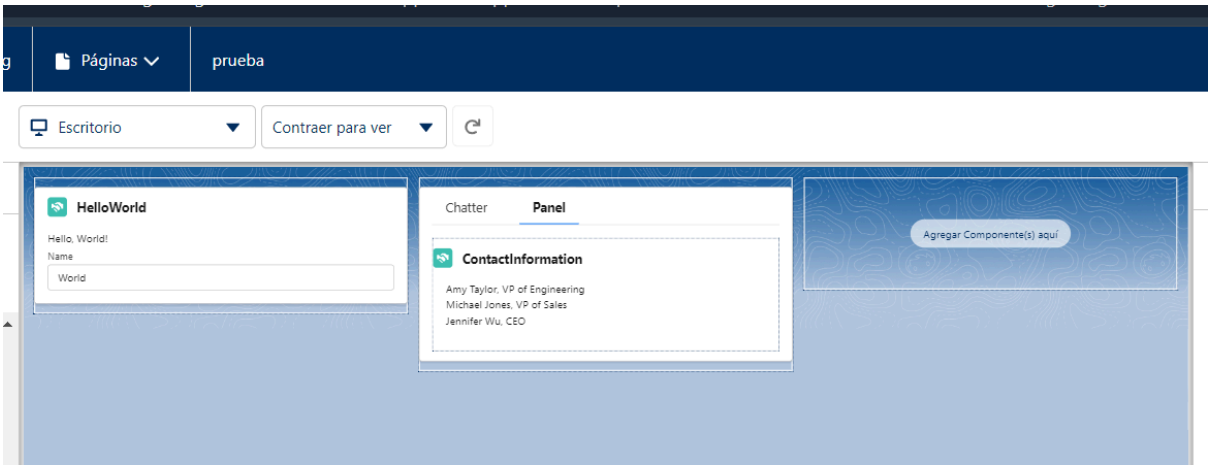
```javascript
import { LightningElement, track } from 'lwc';

export default class MyFirstWebComponent extends LightningElement {
// @track da error si no añades import { track } from 'lwc';
        @track contacts = [
        {
            Id: 1,
            Name: 'Amy Taylor',
            Title: 'VP of Engineering',
        },
        {
            Id: 2,
            Name: 'Michael Jones',
            Title: 'VP of Sales',
        },
        {
            Id: 3,
            Name: 'Jennifer Wu',
            Title: 'CEO',
        },
    ];
```

Bucle for each con los contactos, en el fichero html

```xml
<template>
    <lightning-card title="ContactInformation" icon-name="custom:custom14">
        <div class="slds-m-around_medium">
            <template for:each={contacts} for:item="contact">
                <div key={contact.Id}>
                    {contact.Name}, {contact.Title}
                </div>
            </template>
        </div>
    </lightning-card>
</template>
```

Vista desde Salesforce



TERCER MÓDULO LWC

```
<template>
    <template lwc:if={ready}>
      <div id="display">
        <div>Name: {name}</div>
        <div>Description: {description}</div>
        <div>Category: {category}</div>
        <div>Material: {material}</div>
        <div>Price: {price}</div>
        <div><img src={pictureUrl}/></div>
      </div>
    </template>
    <template lwc:else>
      <div id="waiting">Loading...</div>
    </template>
</template>
```

```
import { LightningElement } from 'lwc';

export default class Apps extends LightningElement {
    name = 'Electra X4';
    description = 'A sweet bike built for comfort.';
    category = 'Mountain';
    material = 'Steel';
    price = '$2,700';
    pictureUrl = 'https://s3-us-west-1.amazonaws.com/sfdc-demo/ebikes/electrax4.jpg';
    ready = false;
    connectedCallback() {
        setTimeout(() => {
            this.ready = true;
        }, 3000);
    }
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
</LightningComponentBundle>
```

Al no poner el targets (<targets>) no se mostrará en el generador de aplicaciones lightning.

LWC Bike, appBike y BikeCard

LWC Bike

```xml
<template>
    <img src={bike.picture} alt="bike picture" />
    <p>{bike.name}</p>
</template>
```

```bike.js
import { LightningElement, api } from 'lwc';

export default class Bike extends LightningElement {
    @api bike;
}
```

LWC appBike

```xml
<template>
    <div>
        <c-bike bike={bike}></c-bike>
    </div>
</template>
```

```appbike.js
import { LightningElement } from 'lwc';

export default class Appbike extends LightningElement {
    bike = {
        name: 'Electra X4',
        picture: 'https://s3-us-west-1.amazonaws.com/sfdc-demo/ebikes/electrax4.jpg'
    };
}
```

LWC BikeCard

El archivo HTML

```
<template>
    <div>
        <div>Name: {name}</div>
        <div>Description: {description}</div>
        <lightning-badge label={material}></lightning-badge>
        <lightning-badge label={category}></lightning-badge>
        <div>Price: {price}</div>
        <div><img src={pictureUrl} /></div>
    </div>
</template>
```

El javaScript

```
import { LightningElement } from 'lwc';

export default class BikeCard extends LightningElement {
    name = 'Electra X4';
    description = 'A sweet bike built for comfort.';
    category = 'Mountain';
    material = 'Steel';
    price = '$2,700';
    pictureUrl = 'https://s3-us-west-1.amazonaws.com/sfdc-demo/ebikes/electrax4.jpg';
}
```

El archivo con las configuraciones necesarias para mostrar el componente en la creación de aplicaciones lightning

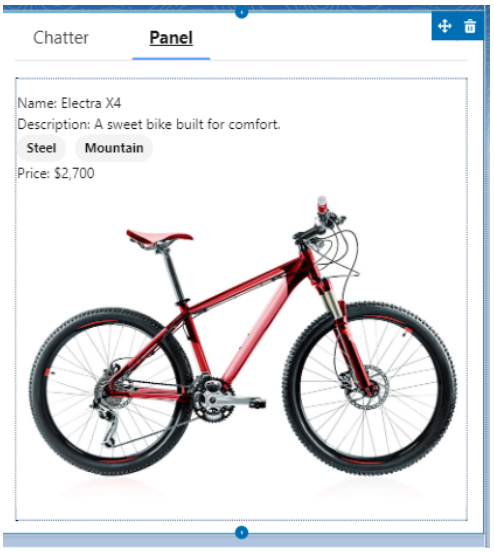
```
p > main > default > lwc > bikeCard > bikeCard.js-meta.xml > LightningComponentBundle
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <masterLabel>Bike Card</masterLabel>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

Se ve como appBike depende de Bike:

Name	Label	Type	Namespace Prefix	Api Version
appbike	appbike	LWC		59
bike	bike	LWC		59

Name	Label
appbike	appbike
bike	bike

Lo que muestra el BikeCard



Seleccionar una bici

LWC Selector

```
<template>
    <div class="wrapper">
    <header class="header">Select a Bike</header>
    <section class="content">
        <div class="columns">
        <main class="main" >
            <c-list onproductselected={handleProductSelected}></c-list>
        </main>
        <aside class="sidebar-second">
            <c-detail product-id={selectedProductId}></c-detail>
        </aside>
        </div>
    </section>
    </div>
</template>
```

```
-meta.xml U    selector.css U ×    deta
> main > default > lwc > selector > selector.c
body {
  margin: 0;
}
.wrapper{
  min-height: 100vh;
  background: #ccc;
  display: flex;
  flex-direction: column;
}
.header, .footer{
  height: 50px;
  background: rgb(255, 255, 255);
  color: rgb(46, 46, 46);
  font-size: x-large;
  padding: 10px;
}
.content {
  display: flex;
  flex: 1;
  background: #999;
  color: #000;
}
.columns{
  display: flex;
  flex:1;
}
.main{
  flex: 1;
  order: 2;
  background: #eee;
}
.sidebar-first{
  width: 20%;
  background: #ccc;
  order: 1;
```

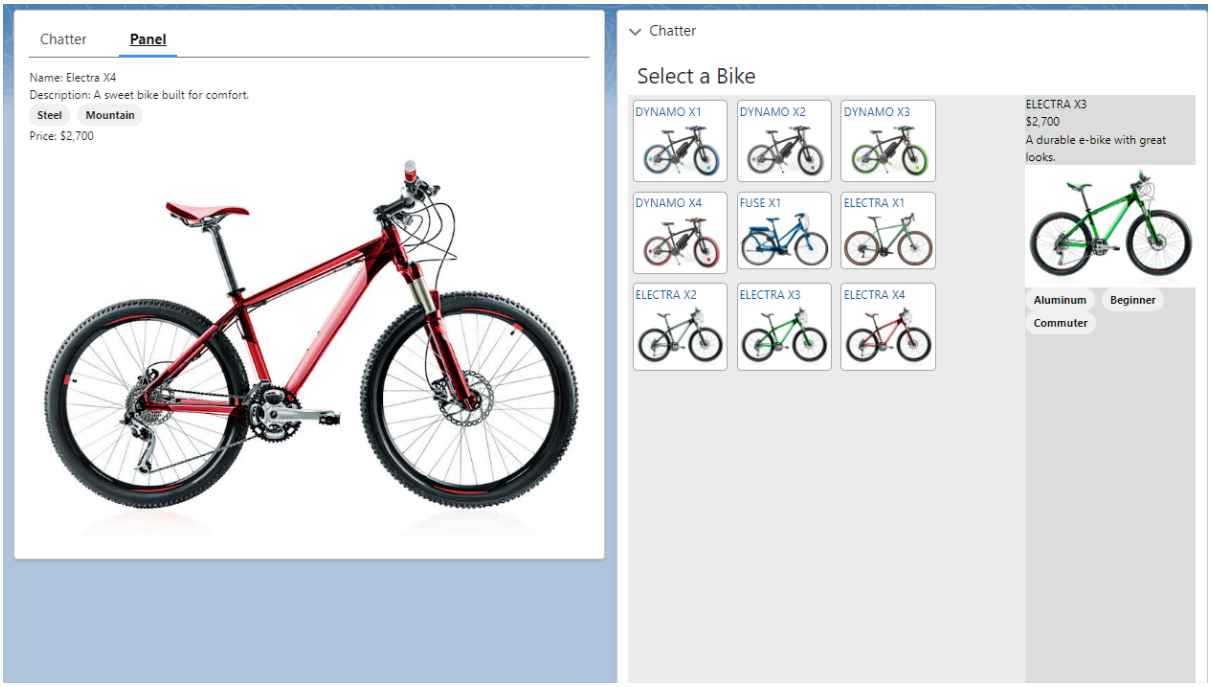
```
import { LightningElement } from 'lwc';

export default class Selector extends LightningElement {
    selectedProductId;

    handleProductSelected(evt) {
        this.selectedProductId = evt.detail;
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

Vista:



Añadimos para mostrar el nombre del usuario

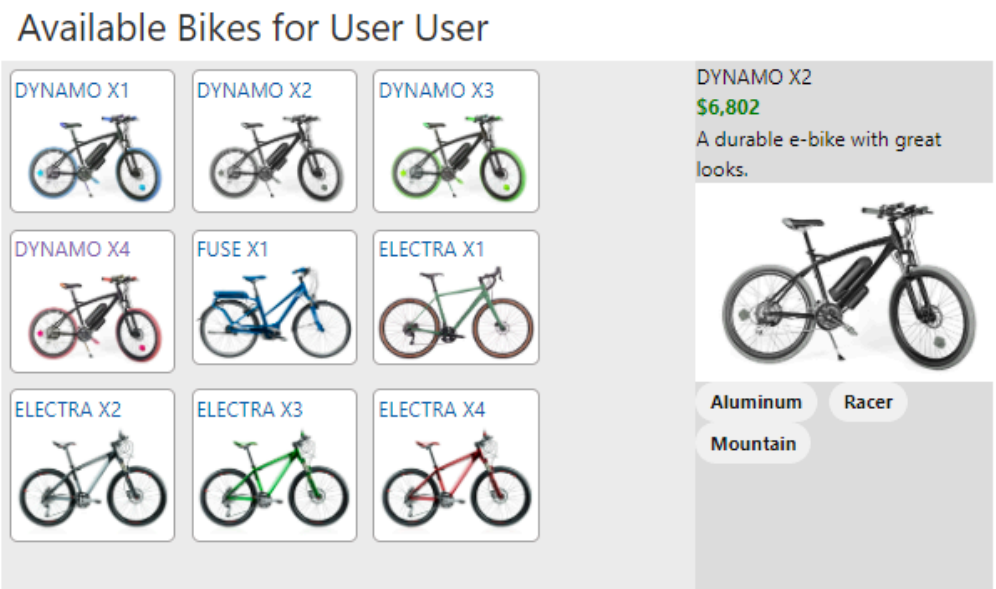
Cambiamos el javaScript

```
import { LightningElement, wire } from 'lwc';
import { getRecord, getFieldValue } from 'lightning/uiRecordApi';
import Id from '@salesforce/user/Id';
import NAME_FIELD from '@salesforce/schema/User.Name';
const fields = [NAME_FIELD];
export default class Selector extends LightningElement {
    selectedProductId;
    handleProductSelected(evt) {
        this.selectedProductId = evt.detail;
    }
    userId = Id;
    @wire(getRecord, { recordId: '$userId', fields })
    user;
    get name() {
        return getFieldValue(this.user.data, NAME_FIELD);
    }
}
```

Cambiamos el Html

```
<template>
    <div class="wrapper">
        <header class="header">Available Bikes for {name}</header>
        <section class="content">
            <div class="columns">
```

Vista:



CUARTO MÓDULO LWC

Los archivos '.js-meta.xml' serán:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
    </targets>
</LightningComponentBundle>
```

LWC AccountCreator

HTML

```html
<template>
    <lightning-card>
        <lightning-record-form
            object-api-name={objectApiName}
            fields={fields}
            onsuccess={handleSuccess}>
        </lightning-record-form>
    </lightning-card>
</template>
```

JavaScript

```javascript
import { LightningElement } from 'lwc';
Alejandro ShowToastEvent } from 'lightning/platformShowToastEvent';
import ACCOUNT_OBJECT from '@salesforce/schema/Account';
import NAME_FIELD from '@salesforce/schema/Account.Name';
import REVENUE_FIELD from '@salesforce/schema/Account.AnnualRevenue';
import INDUSTRY_FIELD from '@salesforce/schema/Account.Industry';
export default class AccountCreator extends LightningElement {
    objectApiName = ACCOUNT_OBJECT;
    fields = [NAME_FIELD, REVENUE_FIELD, INDUSTRY_FIELD];
    handleSuccess(event) {
        const toastEvent = new ShowToastEvent({
            title: "Account created",
            message: "Record ID: " + event.detail.id,
            variant: "success"
        });
        this.dispatchEvent(toastEvent);
    }
}
```

LWC wireGetRecordProperty / wireGetRecordFunction

JavaScript

```javascript
import { LightningElement, api, wire } from 'lwc';
import { getRecord, getFieldValue } from 'lightning/uiRecordApi';
import ACCOUNT_NAME_FIELD from '@salesforce/schema/Account.Name';
export default class WireGetRecordProperty extends LightningElement {
    @api recordId;
    @wire(getRecord, { recordId: '$recordId', fields: [ACCOUNT_NAME_FIELD] })
    account;
    get name() {
        return getFieldValue(this.account.data, ACCOUNT_NAME_FIELD);
    }
}
```

HTML

```html
<template>
    Account Name: {name}
</template>
```

JavaScript

```javascript
import { LightningElement, api, wire } from 'lwc';
import { getRecord, getFieldValue } from 'lightning/uiRecordApi';
import ACCOUNT_NAME_FIELD from '@salesforce/schema/Account.Name';
export default class WireGetRecord extends LightningElement {
    @api recordId;
    data;
    error;
    @wire(getRecord, { recordId: '$recordId', fields: [ACCOUNT_NAME_FIELD] })
    wiredAccount({data, error}) {
        console.log('Execute logic each time a new value is provisioned');
        if (data) {
            this.data = data;
            this.error = undefined;
        } else if (error) {
            this.error = error;
            this.data = undefined;
        }
    }
    get name() {
        return getFieldValue(this.data, ACCOUNT_NAME_FIELD);
    }
}
```

LWC IdsCreateRecord

JavaScript

```
> main > default > lwc > ldsCreateRecord > JS ldsCreateRecord.js > ...
import { LightningElement} from 'lwc';
import { createRecord } from 'lightning/uiRecordApi';
import ACCOUNT_OBJECT from '@salesforce/schema/Account';
import ACCOUNT_NAME_FIELD from '@salesforce/schema/Account.Name';
export default class LdsCreateRecord extends LightningElement {
    handleButtonClick() {
        const recordInput = {
            apiName: ACCOUNT_OBJECT.objectApiName,
            fields: {
                [ACCOUNT_NAME_FIELD.fieldApiName] : 'ACME'
            }
        };
        createRecord(recordInput)
            .then(account => {
                // code to execute if create operation is successful
            })
            .catch(error => {
                // code to execute if create operation is not successful
            });
    }
}
```

Apex ContactController

```
public with sharing class ContactController {

    @AuraEnabled(cacheable=true)
    public static List<Contact> getContactsBornAfter(Date birthDate) {
        return [
            SELECT Name, Title, Email, Phone
            FROM Contact
            WHERE Birthdate > :birthDate
            WITH SECURITY_ENFORCED
        ];
    }
}
```

LWC wireApexProperty

JavaScript

```
import { LightningElement, api, wire } from 'lwc';
import getContactsBornAfter from '@salesforce/apex/ContactController.getContactsBornAfter';
export default class WireApexProperty extends LightningElement {
    @api minBirthDate;
    @wire(getContactsBornAfter, { birthDate: '$minBirthDate' })
    contacts;
    get errors() {
        return (this.contacts.error) ?
            reduceErrors(this.contacts.error) : [];
    }
}
```

LWC callApexImperative

JavaScript

```javascript
import { LightningElement, api, wire } from 'lwc';
import getContactsBornAfter from '@salesforce/apex/ContactController.getContactsBornAfter';
export default class CallApexImperative extends LightningElement {
    @api minBirthDate;
    handleButtonClick() {
        getContactsBornAfter({ // LLama a la Clase Apex llamada ContactController
            birthDate: this.minBirthDate
        })
            .then(contacts => {
                // se ejecuta si devuelve codigo 200
            })
            .catch(error => {
                this.errors = reduceErrors(error);
                //Se ejecuta si devuelve codigo de error
            });
    }
}
```

LWC accountList

JavaScript

```
pp > main > default > lwc > accountList > JS accountList.js > AccountList > (get) errors
import { LightningElement, wire } from 'lwc';
import NAME_FIELD from '@salesforce/schema/Account.Name';
import REVENUE_FIELD from '@salesforce/schema/Account.AnnualRevenue';
import INDUSTRY_FIELD from '@salesforce/schema/Account.Industry';
import getAccounts from '@salesforce/apex/AccountController.getAccounts';
import { reduceErrors } from 'c/ldsUtils';
const COLUMNS = [
    { label: 'Account Name', fieldName: NAME_FIELD.fieldApiName, type: 'text' },
    { label: 'Annual Revenue', fieldName: REVENUE_FIELD.fieldApiName, type: 'currency' },
    { label: 'Industry', fieldName: INDUSTRY_FIELD.fieldApiName, type: 'text' }
];
export default class AccountList extends LightningElement {
    columns = COLUMNS;
    @wire(getAccounts)
    accounts;
    get errors() {
        return (this.accounts.error) ?
            reduceErrors(this.accounts.error) : [];
    }
}
```

HTML

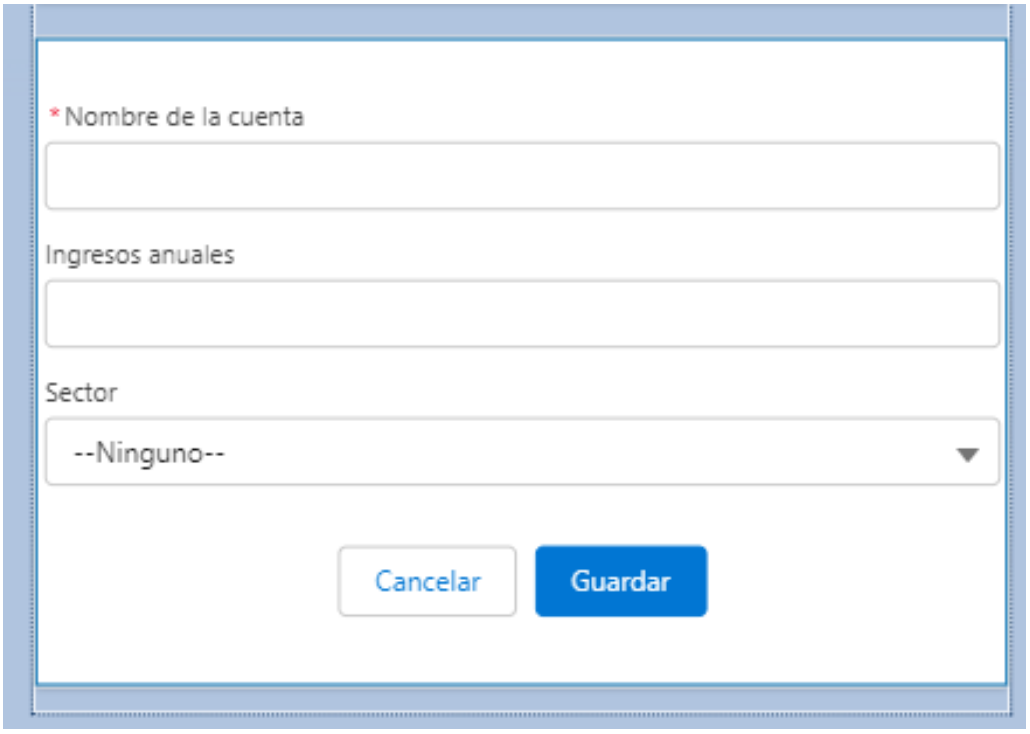
```html
<template>
    <lightning-card>
        <template if:true={accounts.data}>
            <lightning-datatable
                key-field="Id"
                data={accounts.data}
                columns={columns}
            >
            </lightning-datatable>
        </template>
        <template if:true={errors}>
            <p>{errors}</p>
        </template>
    </lightning-card>

</template>
```

Apex AcountController

```
public with sharing class AccountController {
    @AuraEnabled(cacheable=true)
    public static List<Account> getAccounts() {
        return [
            SELECT Name, AnnualRevenue, Industry
            FROM Account
            WITH SECURITY_ENFORCED
            ORDER BY Name
        ];
        // throw new AuraHandledException('Forced error ');
    }
}
```

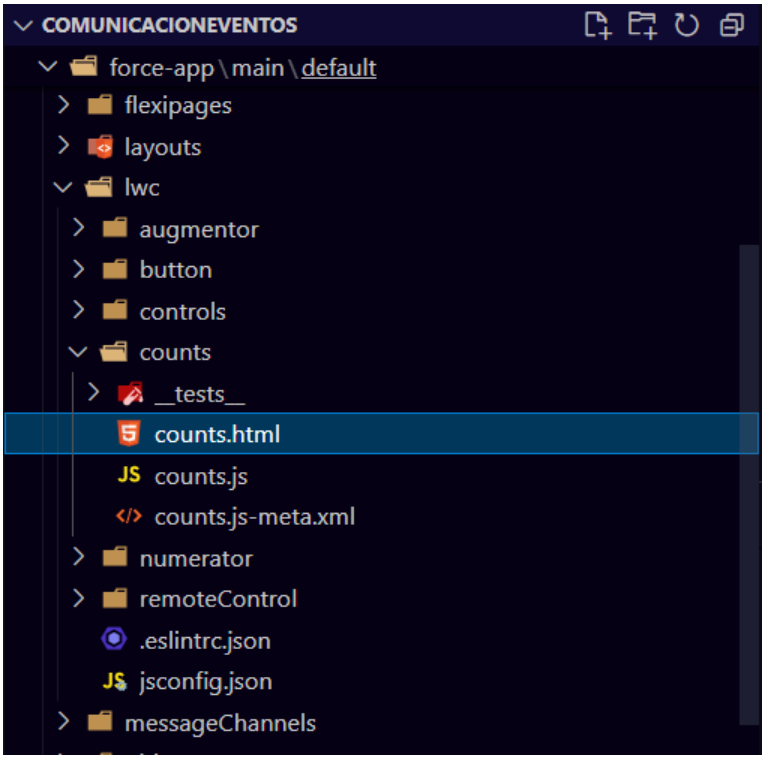
Vista:



Account Name	Annual Revenue	Industry
NuevaCuenta	10.000,00 €	Agriculture
Sample Account for E...		

QUINTO MÓDULO LWC

Para este módulo hemos creado otro proyecto de Salesforce llamado: ComunicacionEventos



LWC Numerator

HTML

```
<template>
    <lightning-card title="Numerator" icon-name="action:manage_perm_sets">
        <p class="slds-text-align_center slds-var-m-vertical_medium">
            Prior Count: <lightning-formatted-number value={priorCount}></lightning-formatted-number>
        </p>
        <p class="slds-text-align_center slds-var-m-vertical_medium">
            Count: <lightning-formatted-number value={counter}></lightning-formatted-number>
        </p>
        <c-controls
            class="slds-show slds-is-relative"
            onadd={handleIncrement}
            onsubtract={handleDecrement}
            onmultiply={handleMultiply}>
        </c-controls>
    </lightning-card>
</template>
```

JavaScript

```javascript
import { LightningElement, api } from 'lwc';

export default class Numerator extends LightningElement {
    // counter = 0;
    //@api counter = 0;

    _currentCount = 0;
    priorCount = 0;
    @api
    get counter() {
      return this._currentCount;
    }
    set counter(value) {
      this.priorCount = this._currentCount;
      this._currentCount = value;
    }

    handleIncrement() {
        this.counter++;
    }
    handleDecrement() {
        this.counter--;
    }
    handleMultiply(event) {
      const factor = event.detail;
      this.counter *= factor;
    }
    @api
    maximizeCounter() {
      this.counter += 1000000;
    }
}
```

LWC Controls
JavaScript

```javascript
import { LightningElement } from 'lwc';

export default class Controls extends LightningElement {
    factors = [0,2,3,4,5,6];
    handleAdd() {
        this.dispatchEvent(new CustomEvent('add'));
      }
      handleSubtract() {
        this.dispatchEvent(new CustomEvent('subtract'));
      }
      handleMultiply(event) {
        const factor = event.target.dataset.factor;
        this.dispatchEvent(new CustomEvent('multiply', {
          detail: factor
        }));
      }
}
```

HTML

```html
<template>
    <lightning-card title="Controls" icon-name="action:upload">
        <lightning-layout>
          <lightning-layout-item flexibility="auto" padding="around-small">
            <lightning-button
              label="Subtract"
              icon-name="utility:dash"
              onclick={handleSubtract}>
            </lightning-button>
          </lightning-layout-item>
          <lightning-layout-item flexibility="auto" padding="around-small" onbuttonclick={handleMultiply}>
            <template for:each={factors} for:item="factor">
              <c-button
                key={factor}
                label={factor}
                data-factor={factor}
                icon="utility:close">
              </c-button>
            </template>
          </lightning-layout-item>
          <lightning-layout-item flexibility="auto" padding="around-small">
            <lightning-button
              label="Add"
              icon-name="utility:add"
              onclick={handleAdd}
              icon-position="right">
            </lightning-button>
          </lightning-layout-item>
        </lightning-layout>
      </lightning-card>
</template>
```

LWC Button

JavaScript

```javascript
import { LightningElement,api } from 'lwc';

export default class Button extends LightningElement {
    @api label;
    @api icon;
    handleButton(event) {
        this.dispatchEvent(new CustomEvent('buttonclick',{
        bubbles: true// Se necesita para que funcione el multiplicador
        }));
    }
}
```

HTML

```html
<template>
    <lightning-button
    label={label}
    data-factor={label}
    icon-name={icon}
    onclick={handleButton}>
    </lightning-button>
</template>
```

LWC Augmentor

HTML

```html
<template>
    <lightning-card title="Augmentor" icon-name="action:download">
        <lightning-layout>
         <lightning-layout-item flexibility="auto" padding="around-small">
            <lightning-input
              label="Set Starting Counter"
              type="number"
              min="0"
              max="1000000"
              value={startCounter}
              onchange={handleStartChange}>
              <lightning-button
              class="slds-var-p-vertical_xx-small"
              label="Add 1m To Counter"
              onclick={handleMaximizeCounter}>
             </lightning-button>
            </lightning-input>
            <lightning-button
          class="slds-var-p-vertical_xx-small"
          label="Add 1m To Counter"
          onclick={handleMaximizeCounter}>
        </lightning-button>
          </lightning-layout-item>
        </lightning-layout>
        <c-numerator
          class="slds-show slds-is-relative"
          counter={startCounter}>
        </c-numerator>
      </lightning-card>
</template>
```

JavaScript

```javascript
import { LightningElement } from 'lwc';

export default class Augmentor extends LightningElement {
    startCounter = 0;
    handleStartChange(event) {
        this.startCounter = parseInt(event.target.value);
    }
    handleMaximizeCounter() {
        this.template.querySelector('c-numerator').maximizeCounter();
    }
}
```

Creamos una carpeta en /default llamada messageChannels que contiene este archivo llamado: **Count_Updated.messageChannel-meta.xml**

```
    <masterLabel>CountUpdated</masterLabel>
    <isExposed>true</isExposed>
    <description>Message Channel to pass Count updates</description>
    <lightningMessageFields>
        <fieldName>operator</fieldName>
        <description>This is the operator type of the manipulation</description>
    </lightningMessageFields>
    <lightningMessageFields>
        <fieldName>constant</fieldName>
        <description>This is the number for the manipulation</description>
    </lightningMessageFields>
</LightningMessageChannel>
```

LWC remoteControl
JavaScript

```
import { LightningElement, wire } from 'lwc';
import { publish, MessageContext } from 'lightning/messageService';
import COUNT_UPDATED_CHANNEL from '@salesforce/messageChannel/Count_Updated__c';
export default class RemoteControl extends LightningElement {
  @wire(MessageContext)
  messageContext;
  handleIncrement() {
    // this.counter++;
    const payload = {
      operator: 'add',
      constant: 1
    };
    publish(this.messageContext, COUNT_UPDATED_CHANNEL, payload);
  }
  handleDecrement() {
    // this.counter--;
    const payload = {
      operator: 'subtract',
      constant: 1
    };
    publish(this.messageContext, COUNT_UPDATED_CHANNEL, payload);
  }
  handleMultiply(event) {
    const factor = event.detail;
    // this.counter *= factor;
    const payload = {
      operator: 'multiply',
      constant: factor
    };
    publish(this.messageContext, COUNT_UPDATED_CHANNEL, payload);
  }
}
```

HTML

```
<template>
    <lightning-card title="Remote Control" icon-name="action:change_record_type">
        <c-controls
          class="slds-show slds-is-relative"
          onadd={handleIncrement}
          onsubtract={handleDecrement}
          onmultiply={handleMultiply}>
        </c-controls>
      </lightning-card>
</template>
```

LWC Counts

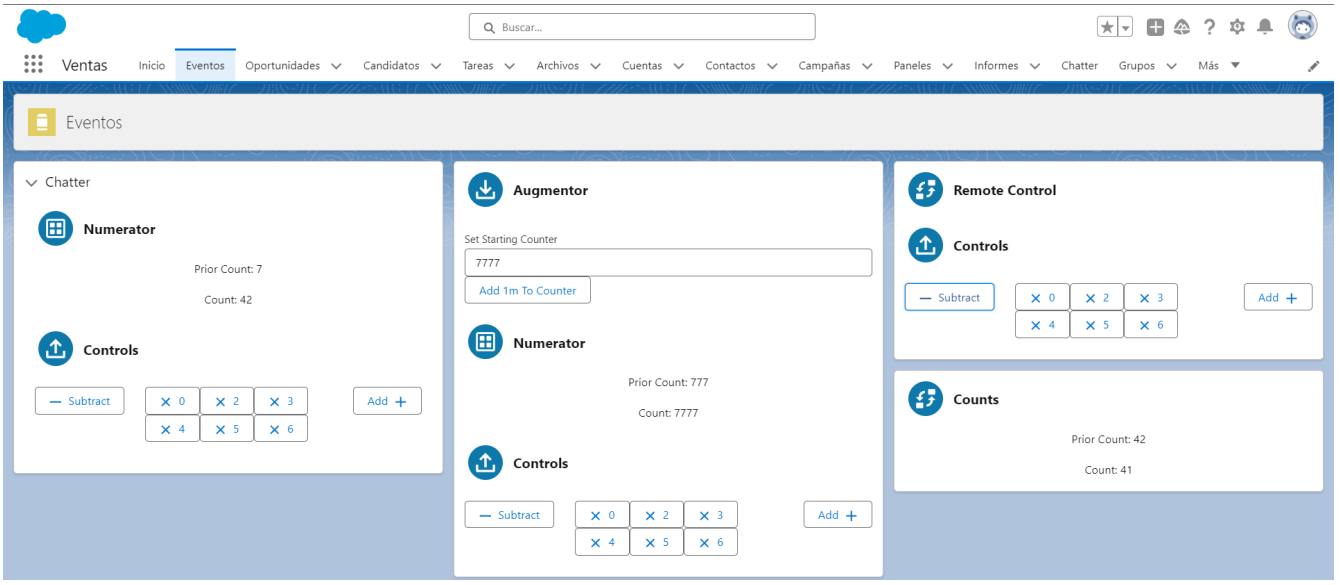
JavaScript

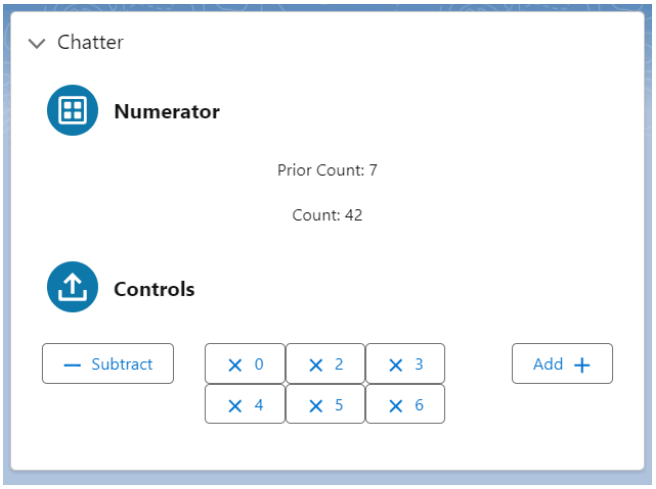
```
> main > default > lwc > counts > JS counts.js > ...
import { LightningElement, wire } from 'lwc';
import { subscribe, MessageContext } from 'lightning/messageService';
import COUNT_UPDATED_CHANNEL from '@salesforce/messageChannel/Count_Updated__c';
export default class Counts extends LightningElement {
  subscription = null;
  priorCount = 0;
  counter = 0;
  @wire(MessageContext)
  messageContext;
  subscribeToMessageChannel() {
    this.subscription = subscribe(
      this.messageContext,
      COUNT_UPDATED_CHANNEL,
      (message) => this.handleMessage(message)
    );
  }
  handleMessage(message) {
    this.priorCount = this.counter;
    if(message.operator == 'add') {
      this.counter += message.constant;
    }else if(message.operator == 'subtract') {
      this.counter -= message.constant;
    } else {
      this.counter *= message.constant;
    }
  }
  connectedCallback() {
    this.subscribeToMessageChannel();
  }
}
```

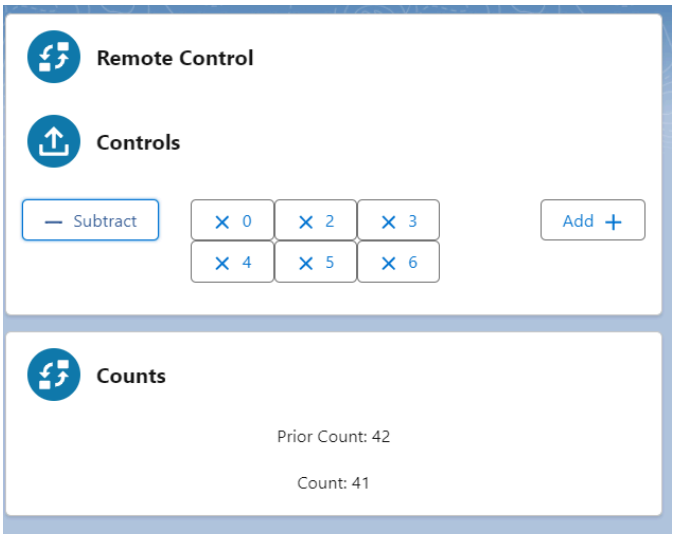
HTML

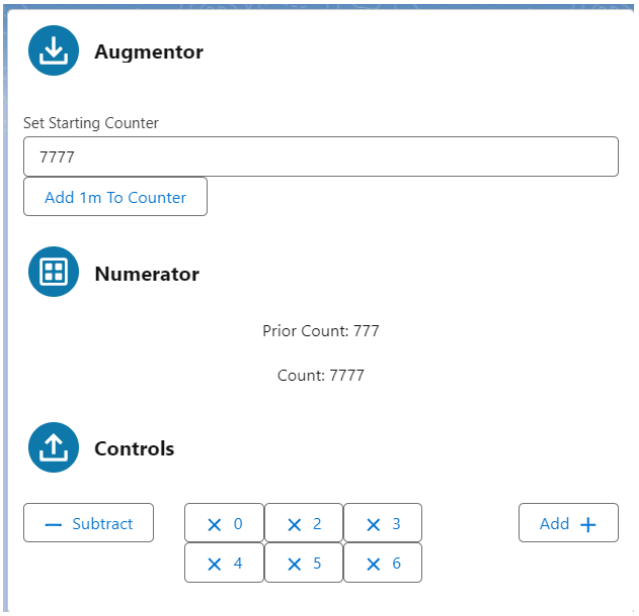
```
<template>
    <lightning-card title="Counts" icon-name="action:change_record_type">
        <p class="slds-text-align_center slds-var-m-vertical_medium">
          Prior Count: <lightning-formatted-number value={priorCount}></lightning-formatted-number>
        </p>
        <p class="slds-text-align_center slds-var-m-vertical_medium">
          Count: <lightning-formatted-number value={counter}></lightning-formatted-number>
        </p>
    </lightning-card>
</template>
```

Vista:


Buscar...
Ventas
Inicio
Eventos
Oportunidades
Candidatos
Tareas
Archivos
Cuentas
Contactos
Campañas
Paneles
Informes
Chatter
Grupos
Más
Eventos
Chatter
Numerator
Prior Count: 7
Count: 42
Controls
Subtract
0
2
3
4
5
6
Add
Augmentor
Set Starting Counter
7777
Add 1m To Counter
Numerator
Prior Count: 777
Count: 7777
Controls
Subtract
0
2
3
4
5
6
Add
Remote Control
Controls
Subtract
0
2
3
4
5
6
Add
Counts
Prior Count: 42
Count: 41


Chatter
Numerator
Prior Count: 7
Count: 42
Controls
Subtract
0
2
3
4
5
6
Add


Remote Control
Controls
Subtract
0
2
3
4
5
6
Add
Counts
Prior Count: 42
Count: 41


Augmentor
Set Starting Counter
7777
Add 1m To Counter
Numerator
Prior Count: 777
Count: 7777
Controls
Subtract
0
2
3
4
5
6
Add

FIN