Spécifications techniques de l'application Salesforce LTP

1. Introduction

Dans le cadre de l’amélioration de l’efficacité opérationnelle de la société *Le Temps des Papillons* (LTP), une application Salesforce Lightning est en cours de conception pour les équipes commerciales et les agents de support client. Cette application vise à offrir une meilleure visibilité sur les données de livraison, une meilleure gestion des opportunités commerciales, et une réduction significative des appels entrants.

---

2. Objectifs de l'application

Fonctionnels :

- Réduction de 70% des appels clients via l'automatisation
- Permettre aux **commerciaux** de gérer efficacement les leads, comptes, contacts, opportunités et livraisons
- Permettre aux **agents de support** d’accéder et de suivre les livraisons par zone géographique (France, Europe, International)
- Offrir une **vision 360° du client**
- S’interfacer avec les systèmes des transporteurs

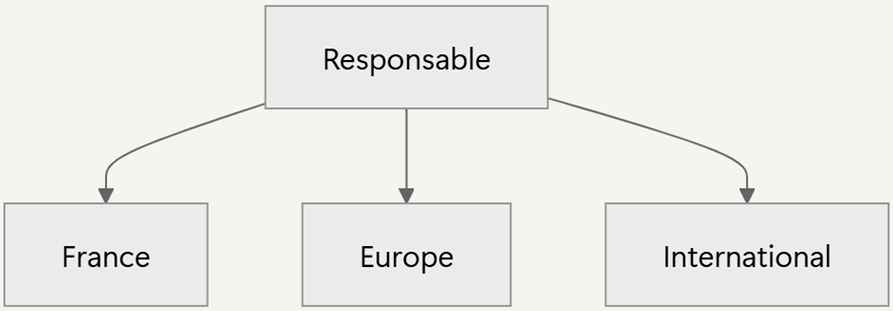
Techniques :

```
flowchart TD
  A[LTP France] -->|Webhook| B[Endpoint Apex REST]
  C[Transport Luxe Europe] -->|Webhook| B
  D[Rapid International] -->|CSV FTP| E[Talend] -->|Bulk API| F[(Salesforce)]
  B -->|Platform Event| G[Flow] --> F
  F --> H[Lien de Suivi Client]
```

> 🧩 User Stories:
>
> - *En tant qu'agent support, je veux accéder rapidement aux livraisons de ma zone.*
> - *En tant que commercial, je veux créer une livraison depuis une opportunité sans ressaisir les infos client.*

---

3. Utilisateurs, rôles et profils



Profil Salesforce	Rôle métier	Description
Commercial	Ventes	Gère les leads, comptes, contacts, opportunités et livraisons
Support Agent	Support (France, Europe, International)	Visualise les clients et modifie les livraisons de sa zone uniquement
Administrateur	Admin technique	Accès complet, maintenance, paramétrage

> 🎯 Tous les agents support partagent le même profil, mais sont différenciés par rôle et règle de partage sur la zone.

---

4. Fonctionnalités par type d'utilisateur

4.1 Commerciaux

- Création et modification de `Lead`, `Account`, `Contact`, `Opportunity`
- Conversion automatisée `Lead → Account + Contact + Opportunity`
- Ajout de `Livraison__c` directement depuis une opportunité
- Visualisation du catalogue produit (`Product2`, `PricebookEntry`)

4.2 Agents Support

- Lecture sur tous les objets clients
- Modification des `Livraison__c` de leur zone uniquement (`Zone__c`)
- Bouton "Actualiser le statut" :

```
// Optimisation du Queueable avec gestion des limites
public class RefreshStatusQueueable implements Queueable, Databas
e.AllowsCallouts {
    private String trackingNum;
     private Integer attempt = 1;
    public void execute(QueueableContext ctx) {
        if (Limits.getCallouts() < Limits.getLimitCallouts()) {
            // Logique callout
        } else {
            System.enqueueJob(this); // Reprise automatique
        }

    }
}
```

- Suivi des statuts de livraison via :
    - API REST pour **LTP France** et **Transport Luxe Europe**

- CSV toutes les 2h pour **Rapid International Transport**

4.3 Clients finaux

- Notifications automatiques :
  - Email à chaque changement de statut
  - SMS optionnel via Twilio (v2)
- Réception d'un **email avec lien de suivi** basé sur `Tracking_Number__c`
- Accès direct à l'URL de tracking, par ex :

```
https://api.transport-france.fr/track?ref={!Livraison__c.Tracking_Number__c}
```

---

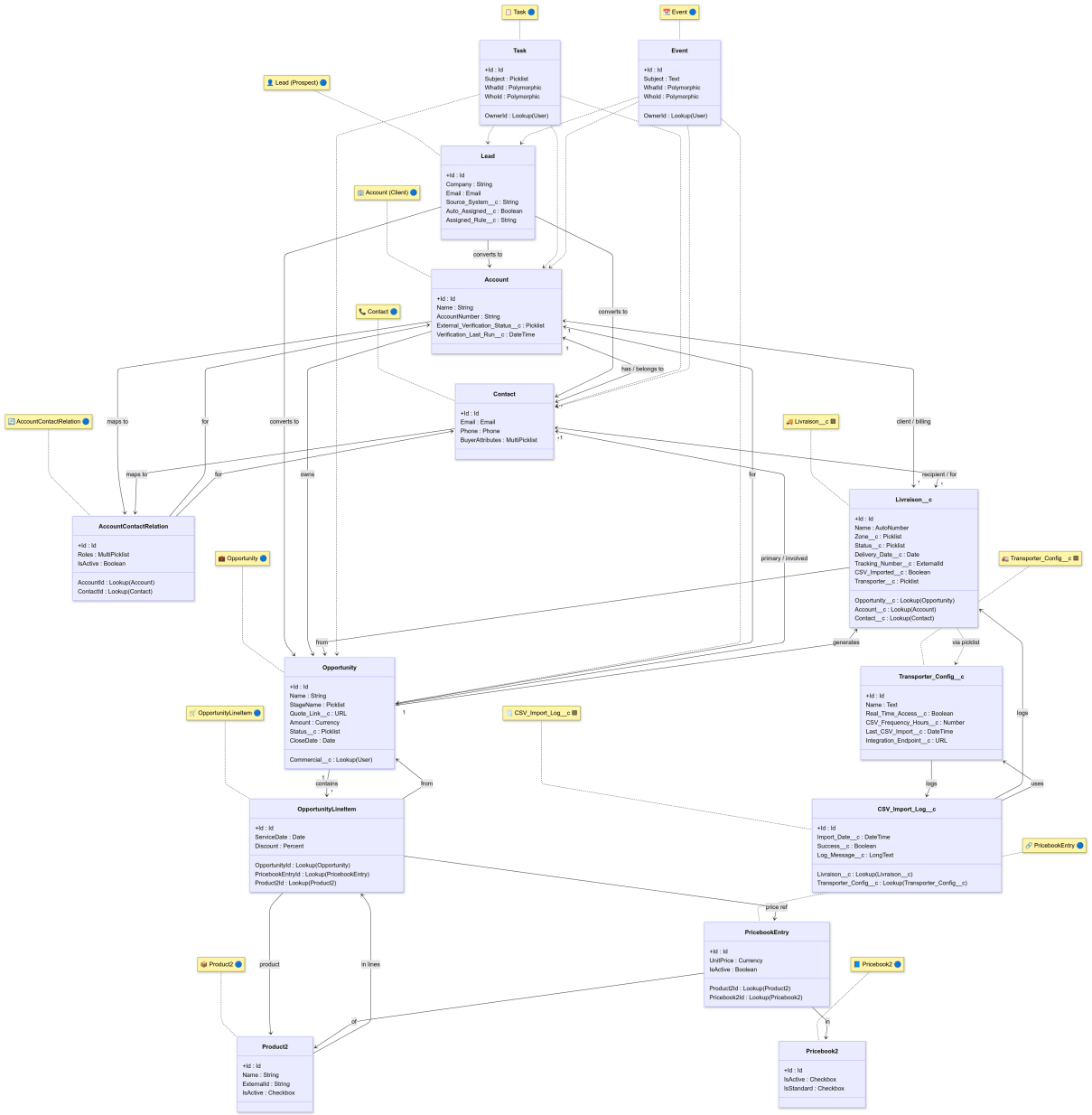
5. Objets Salesforce impliqués

5.1 Standards

Objet	Rôle principal
`Lead`	Prospects à convertir
`Account`	Clients (B2B)
`Contact`	Interlocuteurs des comptes
`Opportunity`	Dossiers commerciaux
`Product2` , `Pricebook2` , `PricebookEntry`	Catalogue produits & tarification
`OpportunityLineItem`	Produits vendus dans les opportunités
`Task` , `Event`	Activités associées
`AccountContactRelation`	Lien N:N entre comptes et contacts

5.2 Objets personnalisés

Objet	Description
`Livraison__c`	Livraison liée à une opportunité
`Transporter_Config__c`	Paramétrage technique des transporteurs (API, CSV, fréquence)
CSV_Import_Log__c	Nouvel objet pour tracer les imports Talend (champs : Batch_ID__c, Statut__c, Message_Erreur__c).



6. Intégration des transporteurs

Tableau des intégrations

Transporteur	Méthode d'intégration	Fréquence	Détails techniques	Solution recommandée
LTP France	API REST temps réel	Immédiat	- Webhook Salesforce- Named Credential- Platform Events	Queueable Apex + Flow

Transporteur	Méthode d'intégration	Fréquence	Détails techniques	Solution recommandée
Transport Luxe Europe	API REST temps réel	Immédiat	- Webhook Salesforce- Named Credential- Platform Events	Queueable Apex + Flow
Rapid International Transport	Fichier CSV	Toutes les 2h	- SFTP sécurisé- Talend ETL- Bulk API v2	Talend + Bulk API v2

NB : Le **Batch Apex seul ne peut pas accéder à un serveur SFTP** → usage interdit ici.

Workbench = **test uniquement**, pas outil d'intégration.

Diagramme d'authentification Talend-Salesforce

```
sequenceDiagram
    participant Talend
    participant Salesforce
    Talend->>Salesforce: Authentification (Named Credentials OAuth2)
    Salesforce-->>Talend: Token d'accès valide
    Talend->>Salesforce: Envoi des données via Bulk API v2
    Salesforce-->>Talend: Confirmation de réception
```

Points clés:

1. **Webhooks**:
    - Configurer des endpoints REST sécurisés dans Salesforce
    - Utiliser des Platform Events pour découpler le traitement
    - Exemple de code:

```
@RestResource(urlMapping='/delivery-webhook')
global class DeliveryWebhook {
    @HttpPost
    global static void handleUpdate() {
     if (!RestContext.request.headers.containsKey('X-SFDC-CSRF-Protec
```

```
tion')) {
            RestContext.response.statusCode = 403; // Forbidden
            return;
        }
        // Traitement du payload
        EventBus.publish(new DeliveryEvent__e(
            TrackingNumber__c = trackingNum,
            Status__c = status
        ));
    }
}
```

2. **Intégration Talend**:
   - Configuration optimale du job Talend:

```
// Configuration du composant tSalesforceOutputBulkExec
tSalesforceOutputBulkExec_1.setExternalIdField("Tracking_Number__c");
tSalesforceOutputBulkExec_1.setBatchSize(10000);
tSalesforceOutputBulkExec_1.setUpsert(true);
tSalesforceOutputBulkExec_1.setConnectionTimeout(60000); // Évite les timeouts
```

3. **Sécurité**:
   - Remote Site Settings obligatoires pour tous les endpoints
   - Named Credentials recommandés pour stocker les identifiants
   - Vérification FLS/CRUD dans tous les appels Apex

Recommandations supplémentaires:

- Pour les tests: Implémenter des `HttpCalloutMock`
- Monitoring: Créer un objet `Integration_Log__c` pour tracer tous les appels
- Gestion des erreurs: Mettre en place une stratégie de reprise (retry mechanism)

Exemple de code Apex optimisé:

```
public with sharing class DeliveryService {
    @Future(callout=true)
    public static void refreshStatus(String trackingNumber) {
        try {
            HttpRequest req = new HttpRequest();
            req.setEndpoint('callout:TransportFrance_NC/track/'+trackingNumb
er);
            req.setMethod('GET');

            HttpResponse res = new Http().send(req);

            if(res.getStatusCode() == 200) {
                // Traitement de la réponse
                update new Livraison__c(
                    Tracking_Number__c = trackingNumber,
                    Status__c = parseStatus(res.getBody())
                );
            }
        } catch(Exception e) {
            // Journalisation des erreurs
            ErrorLogger.log(e);
        }
    }
}
```

---

7. Architecture technique Apex & Flow

🔷 Architecture multi-couche

Couche	Élément	Rôle
UI	Lightning Page / Email	Bouton "Suivre ma livraison", lien dynamique
Flow	`Record-Triggered` , `Screen Flow`	Notifications, blocages, automatisation, validation
Webservice	`@RestResource` Apex	Réception CSV ligne par ligne

Couche	Élément	Rôle
Service	`TransporterCalloutService`	Appel API GET, parsing JSON
Log	`CSV_Import_Log__c`	Suivi des erreurs import Talend implémenter l'interface `Batchable` pour les reprises automatiques.

> ⚠️ Aucune logique métier dans Trigger — tout est encapsulé dans Flow ou Apex Services.

---

8. Import CSV orchestré par Talend

📂 Fonctionnement global :

```
[Serveur SFTP] → [Talend Job planifié] → [Webservice Apex REST POST li
gne par ligne]
                                    ↓
                        Création Livraison__c via Tracking_Number__c (Externa
l Id)
                                    ↓
                    Journalisation → CSV_Import_Log__c
```

```
graph TB
  A[SFTP] --> B[Talend Job]
    B --> C{{Transformation}}
    C --> D[Bulk API v2]
    D --> E[Upsert Livraison__c]
    E --> F[CSV_Import_Log__c]
    D --> G[Erreurs → S3]
```

Mise en œuvre:

1. Dans Talend:

```
// tSalesforceOutputBulkExec configuration
String bulkJobId = tSalesforceOutputBulkExec_1.createJob(
    "Livraison__c",
    "upsert",
```

```
    "Tracking_Number__c",
    contentStream
);
```

1. Créer un champ `Batch_ID__c` sur `CSV_Import_Log__c` pour tracer les jobs
2. Monitoring:

```
SELECT COUNT(Id), Status__c
FROM CSV_Import_Log__c
WHERE Batch_ID__c = 'job123'
GROUP BY Status__c
```

✅ Avantages :

- **Respects governor limits**
- Contrôle ligne à ligne via `try/catch`
- Détection automatique des doublons via `External Id`
- Sécurité assurée : authentification, champs protégés (`CSV_Imported__c`)

---

9. Appel API REST – Suivi en temps réel

📦 Fonction : `fetchDeliveryStatus(trackingNumber)`

Élément	Détail
Méthode	`GET /track?ref={Tracking_Number__c}`
Authentification	Header Bearer Token ou Named Credential
Consommateur	Apex (`TransporterCalloutService`) ou bouton Lightning
Réponse	JSON (`status`, `ETA`, etc.)
Affichage	Modal, toast ou champ `Status__c` mis à jour
Simulation test	`HttpCalloutMock`

Composant Apex	Type	Description
`TransporterCalloutService`	`Service`	Classe centrale d'appel HTTP REST GET
`TransporterCalloutMock`	`HttpCalloutMock`	Simulation de réponse API en test

`Transporter_Config__c`	`Custom Object`	Stockage des endpoints/API keys des transporteurs
`Tracking_Number__c`	`Champ externe`	Utilisé comme clé de requête GET `/tracking?ref=...`

🔷 Sécurité & configuration

Élément Salesforce	Rôle	Exemple
`Remote Site Settings`	Autoriser les URLs distantes (obligatoire)	`https://api.ltp-france.fr`
`Named Credential` (optionnel)	Authentification sécurisée (OAuth/Basic)	`TransportFrance_NC`
`with sharing` + `FLS/CRUD`	Vérification dans chaque méthode	`Schema.sObjectType...isAccessible()`

🔷 Exemple simplifié de code Apex (GET)

```
public class TransporterCalloutService {
    @future(callout=true)
    public static void fetchDeliveryStatus(String trackingNumber) {

        HttpRequest req = new HttpRequest();
            req.setEndpoint('callout:TransportFrance_NC/track?ref='+trackin
gNum);
            req.setMethod('GET');
        req.setHeader('Content-Type', 'application/json');

        Http http = new Http();
        HttpResponse res = http.send(req);

        if (res.getStatusCode() == 200) {
            Map<String, Object> payload = (Map<String, Object>) JSON.deseri
alizeUntyped(res.getBody());
            // Exemple de traitement...
        } else {
            SEventBus.publish(new Error_Event__e(
            Message__c = 'Erreur '+res.getStatusCode()
        ));
```

```
    }
} catch(Exception e) {
    System.debug('Erreur: '+e.getMessage());
        }
    }
}
```

🔷 Tests unitaires et mock callouts

```
@isTest
global class MockTransporteur implements HttpCalloutMock {
    global HTTPResponse respond(HTTPRequest req) {
        HttpResponse res = new HttpResponse();
        res.setHeader('Content-Type', 'application/json');
        // Scénarios testés :
        // 1. Réponse valide
        // 2. Erreur 500
        // 3. Timeout simulé
        return res;
    }
}
```

🔷 Recommandations techniques

- Toujours encapsuler les appels dans des `@future(callout=true)` ou `Queueable` si enchaînement nécessaire.
- Prévoir une **gestion des erreurs** robuste : timeout, 404, 500.
- Tester tous les cas (réponse valide, invalide, indisponible).
- Isoler les URLs/API Keys dans des **Custom Metadata Types** ou `Transporter_Config__c`.

10. Composant Lightning "Suivre ma livraison"

🔘 Lien / bouton de suivi client

- Un **bouton “Suivre ma livraison”** est accessible dans :

- Email de confirmation
- Lightning Record Page

✅ **Permet au client** ou à l'agent de vérifier la livraison en temps réel

Lieu	Type	Contenu
Page `Livraison__c`	Bouton Lightning	Ouvre l'URL externe avec `Tracking_Number__c`
Email client	Template	Lien de suivi personnalisé
Flow	Notification	Contient le lien de tracking

```
HYPERLINK("https://api.transport-france.fr/track?ref=" & Tracking_Number__c, "Suivre ma livraison")
```

LWC - deliveryTracker :

```
public with sharing class DeliveryController {
    @AuraEnabled
    public static Delivery__c getDelivery(Id recordId) {
        // Vérifie l'accès en lecture au champ Status__c
        if (!Schema.sObjectType.Delivery__c.fields.Status__c.isAccessible()) {
            throw new SecurityException('Accès refusé');
        }
        return [SELECT Status__c, Tracking_Number__c FROM Delivery__c WHER
    }
}
```

```
import { LightningElement, api, wire } from 'lwc';
import refreshStatus from '@salesforce/apex/DeliveryController.refreshStatus

export default class DeliveryTracker extends LightningElement {
    @api recordId;
    @track delivery;
    @track error;

    @wire(getDelivery, { recordId: '$recordId' })
    wiredDelivery({ error, data }) {
        if (data) this.delivery = data;
```

```
        if (error) this.error = error;
    }

    handleRefresh() {
        refreshStatus({ recordId: this.recordId })
            .then(() => { /* Rafraîchir l'UI */ });
    }
}
```

```
<template>
    <lightning-card title="Suivi de livraison">
        <div class="slds-p-around_medium">
            <p>Statut: <lightning-badge>{status}</lightning-badge></p>
            <lightning-button
                label="Actualiser"
                onclick={refreshStatus}>
            </lightning-button>
            <a href={trackingUrl} target="_blank">
                Voir sur le site transporteur
            </a>
        </div>
    </lightning-card>
</template>
```

---

11. Outils et plateformes utilisées

Outil	Rôle
Salesforce CLI	Déploiement
VS Code + extensions	Développement Apex
Talend	Import CSV, transformation, log, orchestration
Salesforce Data Loader	Imports manuels ou tests
Salesforce Workbench	Test API, vérification données
Mulesoft (optionnel)	Centralisation flux si v2

Outil	Rôle
Flow & Validation Rule	Low-code, contrôle UI
Platform Event (⚠️ V2)	Notification push transporteurs (Webhook)

12. Contraintes et gouvernance

Type	Détail
Volumétrie	2M comptes, 3M contacts, 250k opportunités
Sécurité	`with sharing`, FLS/CRUD, champ externe, restriction zone
Tests	Couverture ≥ 85 %, mocks API
Logs	`CSV_Import_Log__c`, logs Talend
Résilience	`try/catch`, gestion erreurs API

13. Roadmap évolutive (version 2)

-
- **Monitoring temps réel** :
  - Dashboard Salesforce avec :
    - Taux de réussite des intégrations
    - Délai moyen de traitement

```
SELECT COUNT(Id), Transporteur__c
FROM CSV_Import_Log__c
GROUP BY Transporteur__c
```

- **Portail client** :
  - Page Lightning **publique** avec :
    - Suivi par `Tracking_Number__c`
    - Chatbot intégré pour le support
- 📦 Packaging complet (metadata API, change sets)

14. Conclusion

L’architecture mise en place :

- ✅ Supprime les points faibles des triggers, batchs, scheduled jobs
- ✅ Utilise des **services encapsulés, testables, sécurisés**
- ✅ Est **compatible avec les contraintes du cloud Salesforce**
- ✅ Permet un **suivi précis et dynamique des livraisons**
- ✅ Offre une **expérience utilisateur claire et efficace** pour le support, les commerciaux, et les clients

> 🔐 Elle respecte les principes clés Salesforce : séparation des responsabilités, idempotence, gouvernance API, sécurité des données.