PLEC DE PRESCRIPCIONS TÈCNIQUES PER A LA CONTRACTACIÓ DEL SERVEI PER EL DESENVOLUPAMENT D'UN GESTOR DE CERTIFICACIONS PER AL PROJECTE INTERNUNIVERSITARI DE GENERACIÓ DE CONTINGUTS PER EL DESENVOLUPAMENT DE COMPETÈNCIES DIGITALS. (Contracte específic de l'E23/43: Sistema Dinàmic d'Adquisició per a l'acreditació de proveïdors de serveis de desenvolupament de programari i de consultoria per a la governança tecnològica (CSUC))

Actuació GCxdCD a UNIDIGITAL C21.I05.P02.S31 (inicialment C21.I05.P1.S8)

EXPEDIENT 2024/131

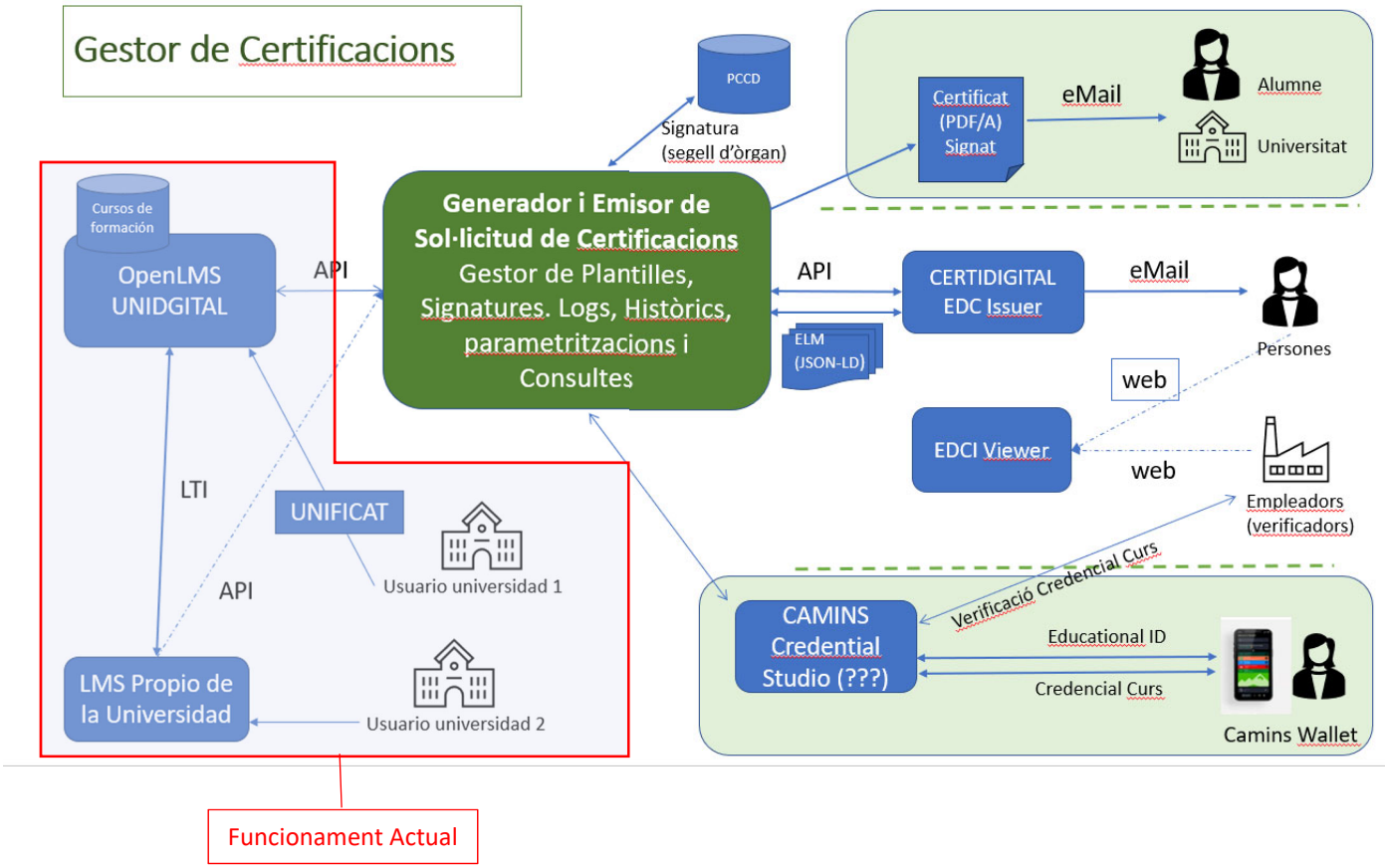
L'adjudicatari ha de definir i fer el plantejament del projecte, que ha de ser aprovat per totes les universitats participants (Universitat de Barcelona, Universitat Autònoma de Barcelona, Universitat Politècnica de Catalunya, Universitat de Girona, Universitat de Lleida i Universitat Rovira i Virgili). El gestor de certificacions es pot basar en diferents models:

- En base al model tradicional: pdfs signats
- En base a la implementació del model d'usuari sobirà en la gestió de les seves dades, basat en EBSI i ús de cartera virtual (i complint el reglament d'Identificació electrònica, autenticació i serveis de confiança EIDAS)

Es vol desenvolupar un **generador i emissor de sol·licituds de certificacions europees** que pugui gestionar l'enviament de sol·licituds de g**eneració de certificacions/acreditacions a diferents destinataris** que generaran certificacions/acreditacions de **diferents tipologies i formats (PDF/A**, credencial per a cartera virtual i credencial en base a format EDC- EuropeanDigital Credential for Learning).

És objecte d'aquest projecte desenvolupar el generador de sol·licituds que ha de:

- Gestionar sol·licituds de credencials/acreditacions a emetre (integració amb Open LMS (CSUC). Aquestes sol·licituds podran arribar periòdicament des de la pròpia plataforma Open LMS o ser demandades des de el gestor de sol·licituds a demanda i segons uns paràmetres desitjats.
- El Generador haurà de permetre usuaris diferents i gestors independents per universitats de forma que segons el rol hi hagi accés a informació i permisos de gestió d'una única universitat o de totes.
- El Generador també haurà de permetre
    - Gestionar plantilles de certificats a fer (en funció de la universitat peticionaria i el col·lectiu del alumne (PTGAS o PDI)
    - Gestionar certificats-electrònics amb els que signar cada "certificat/acreditació" i que pot variar també segons el col·lectiu (PTGAS o PDI) i sempre en funció d ela Universitat
    - Gestionar històric de credencials i certificats/acreditacions emeses
    - Gestió de logs.
    - Incloure funcionalitats per el/els gestors dels generador.
- Integrar-se amb l'emissor de certificacions CERTIDIGITAL a través d'APIs i enviaments de certificats a emetre en format JSON-LD



Actualment s'obtenen de la plataforma Open LMS una sèrie d'informes (a demanda), on consta la relació d'alumnes que han cursat i superat cursos i la universitat a la que pertanyen. A partir d'aquest informe les Universitats generen els certificats corresponents (d'assoliment del curs o competència adquirida). Cada universitat genera els del seus treballadors, i en alguns casos, depenent de la universitat, per a signar aquests certificats es fa servir un certificat electrònic o altre(segells d'òrgan en la majoria dels casos) en funció de si l'alumne és de perfil PTGAS o PDI:

Universitat	Per a PDI	Per a PTGAS
UB	Segell del IDP-ICE (ja existeix)	Actualment a demanda i amb certificat personal (T-CAT). Futur ?
UAB		
UPC	Segell "serveis Eadmin- UPC", EC- Sector Públic de l'AOC	No es certifiquen actualment. Futur ?

UdG		
URV	Segell EC-Sector Públic de l'AOC mitjançant EDCI	Segell EC- Sector Públic de l'AOC mitjançant EDCI
UdL		

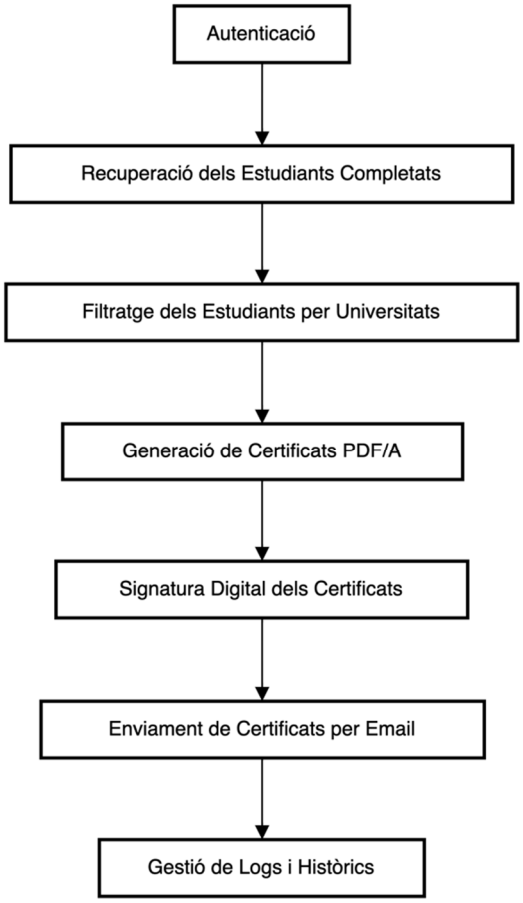
Funcions a Desenvolupar:

Model tradicional i conservador - interacció OpneLMS i generació PDF/A (primer requadre verd clar del esquema)

Es requereix el desenvolupament d'una aplicació que, periòdicament, recuperi els estudiants que han completat els seus cursos a OpenLMS, generi certificats en format PDF/A basats en plantilles definides per cada universitat i signi digitalment els certificats utilitzant els certificats digitals dipositats en RedTrust. Finalment, els certificats seran enviats per correu electrònic als estudiants. Aquest document ha descrit de manera detallada el procés necessari per implementar una solució automàtica per a l'emissió de certificats acadèmics en format PDF/A, la seva signatura digital utilitzant RedTrust i l'enviament per correu electrònic als estudiants. A l'Annex I es proporciona un codi d'exemple que es pot adaptar a les necessitats específiques de cada universitat, assegurant la privacitat i la seguretat de les dades de cada institució i dels seus estudiants.

Aquest mòdul ha de:

- Recuperar periòdicament els estudiants que han completat els cursos a OpenLMS. Aquesta recuperació podrà ser parametritzable i configurable per a cada universitat.
- Generar certificats/acreditacions en format PDF/A basats en plantilles específiques per a cada universitat, curs i perfil dels alumnes.
- Signar digitalment els certificats/acreditacions utilitzant els certificats digitals (també en funció de la universitat i perfil del alumne) dipositats en RedTrust o en la plataforma de custodia de certificats que proporciona el CSUC PCCD.
- Enviar els certificats/acreditacions signats per correu electrònic als interessats (i si s'escau, a les universitats corresponents els mateixos certificats/acreditacions o informació dels certificats emesos).
- Gestionar històrics i logs, consultes d'enviaments a certificació i històrics de manera segura per a múltiples universitats.



Model Wallet CAMINS - interacció OpneLMS i CAMINS EBSI (segon requadre verd clar del esquema)

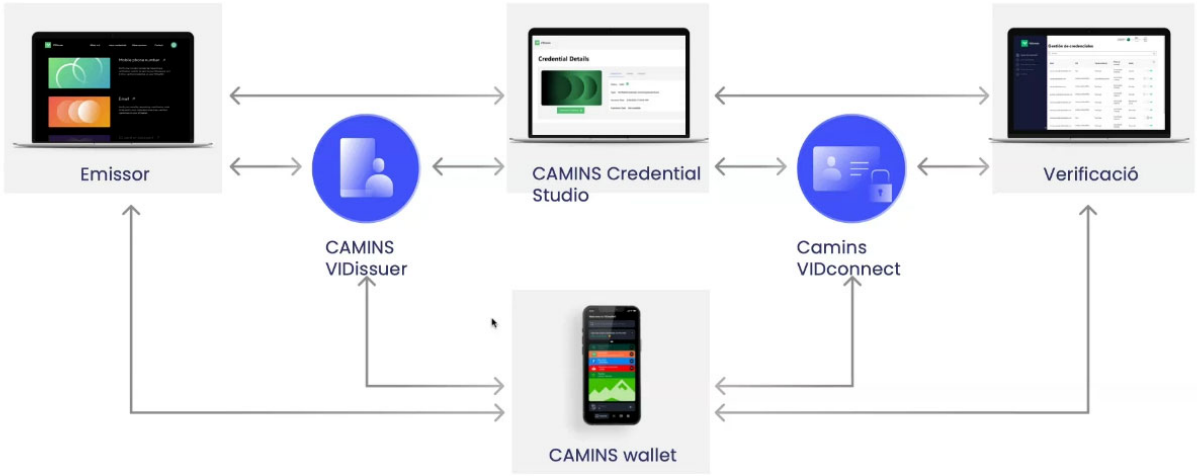
Aquesta secció descriu el conjunt de tasques necessàries per al desenvolupament d'un mòdul o aplicació basada en la filosofia de Self Sovereign Identity (SSI) i els estàndards de W3C, implementats a través de la European Blockchain Service Infrastructure (EBSI). Suposa la possibilitat de que sigui el propi interessat (alumne) qui demani la seva certificació/Acreditació/Credencial a la entitat emissora d ela mateixa (la universitat a la que pertany), de forma que la pugui tenir en una cartera virtual i presentar en altres organismes (inicialment Universitats) des del seu Wallet. L'aplicació permetrà que, a petició de l'usuari, es generin i validin credencials verificables d'identitat i acadèmiques, assegurant la privacitat i la seguretat de les dades.

Aquesta opció ha de partir del codi obert de CAMINS (UNIDIGITAL), en la seva modalitat basada en EBSI (European Blockchain Services Infraestructure) que, entre altres, implementa una funcionalitat per fer la petició i verificació d'una credencial verificable. Caldrà desenvolupar un portal i una aplicació que demani una credencial verificable d'identitat (e.g. educationalID), i un cop verificada, faci un procés d'identity matching per cercar, al

contenidor de l'OpenLMS d'assoliments educatius i, per la persona identificada (PTGAS, PDI, Estudiant), pressenti aquells assoliments susceptibles de ser certificats/acreditats. Un cop seleccionats i validada la petició, caldrà fer l'emissió al wallet de l'interessat.

Aquest mòdul ha de:

- Permetre als usuaris sol·licitar una acreditació fent un "click" en el servei "Requerir acreditació".
- Generar una petició de credencial verificable de identitat "EducationalID" mitjançant la EBSI (ja fet i funcionant al projecte CAMINS – UNIDIGITAL).
- Validar les presentacions verificables compartides pels usuaris (VP).
- Utilitzar l'atribut "ABC" de la credencial "EducationalID" validada per recuperar els logros acadèmics, que en aquest cas seran els cursos assolits de competències digitals en el OpenLMS.
- Permetre als usuaris seleccionar els logros a acreditar (els de Open LMS, com a mínim) i generi les acreditacions corresponents.
- Notificar la disponibilitat de les acreditacions al wallet de l'usuari.



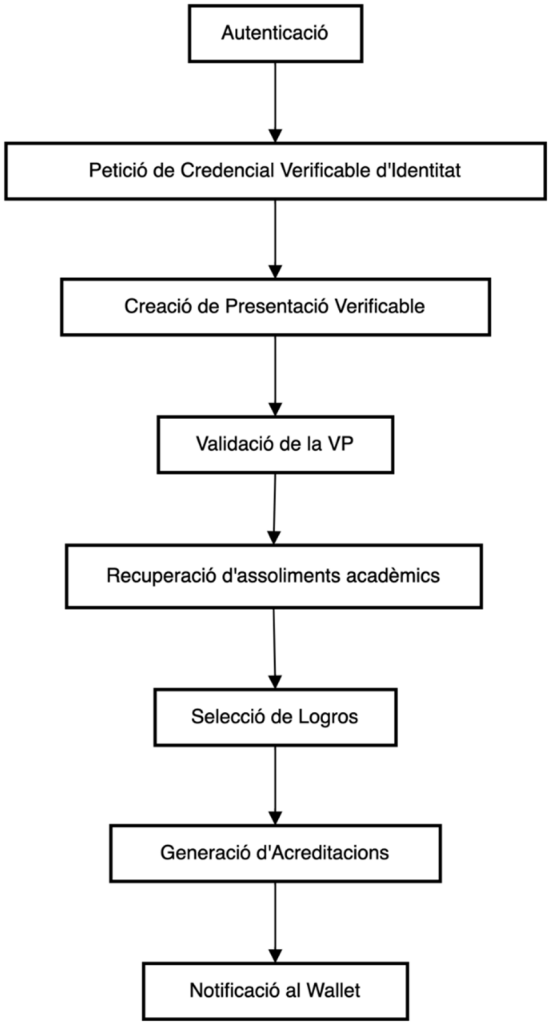
On l'emissor serà la Universitat a la que pertany l'usuari, i el Verificador inicialment seran les pròpies universitats participants en el projecte Camins (UB, UAB, UPC, URV i UdG).

Es tracta d'un nou cas d'ús dintre del projecte CAMINS - EBSI .

Passos

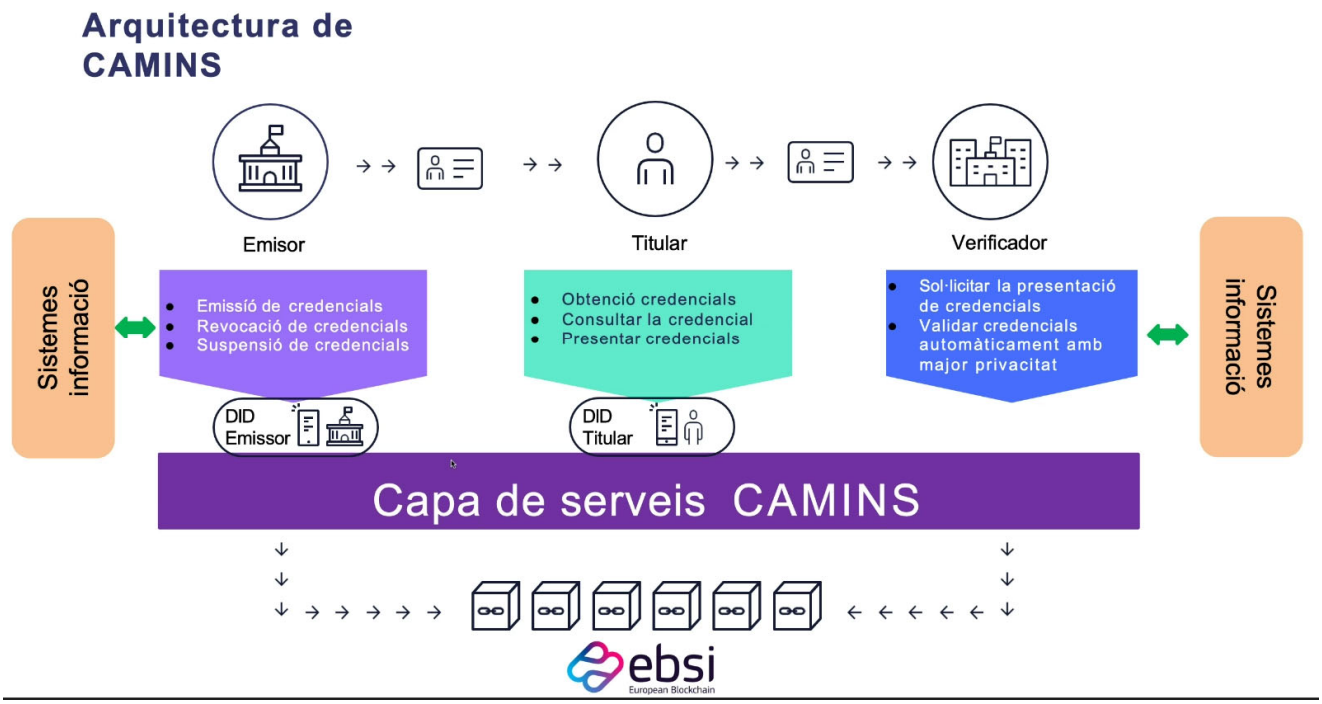
1. Autenticació i Obtenció de Token: Autenticar-se amb els serveis de EBSI i obtenir un token JWT.
2. Petició de Credencial Verificable d'Identitat: Generar una sol·licitud per a la credencial "EducationalID" i enviar-la a l'usuari per a la seva acceptació.
3. Creació de Presentació Verificable (VP): Després de l'acceptació per part de l'usuari, crear una VP que contingui el "EducationalID" i compartir-la amb el servei.
4. Validació de la VP: Validar la presentació verificable (integritat, emissor, estat, etc).

5. Recuperació de Logros Acadèmics: Utilitzar l'atribut "ABC" de la credencial "EducationalID" per recuperar els logros acadèmics associats.
6. Selecció de Logros: Permetre a l'usuari seleccionar els logros que desitja acreditar.
7. Generació d'Acreditacions: Generar les acreditacions seleccionades seguint el format ELMv3.2.
8. Notificació al Wallet: Notificar al wallet de l'usuari la disponibilitat de les acreditacions per a la seva acceptació i emmagatzematge.



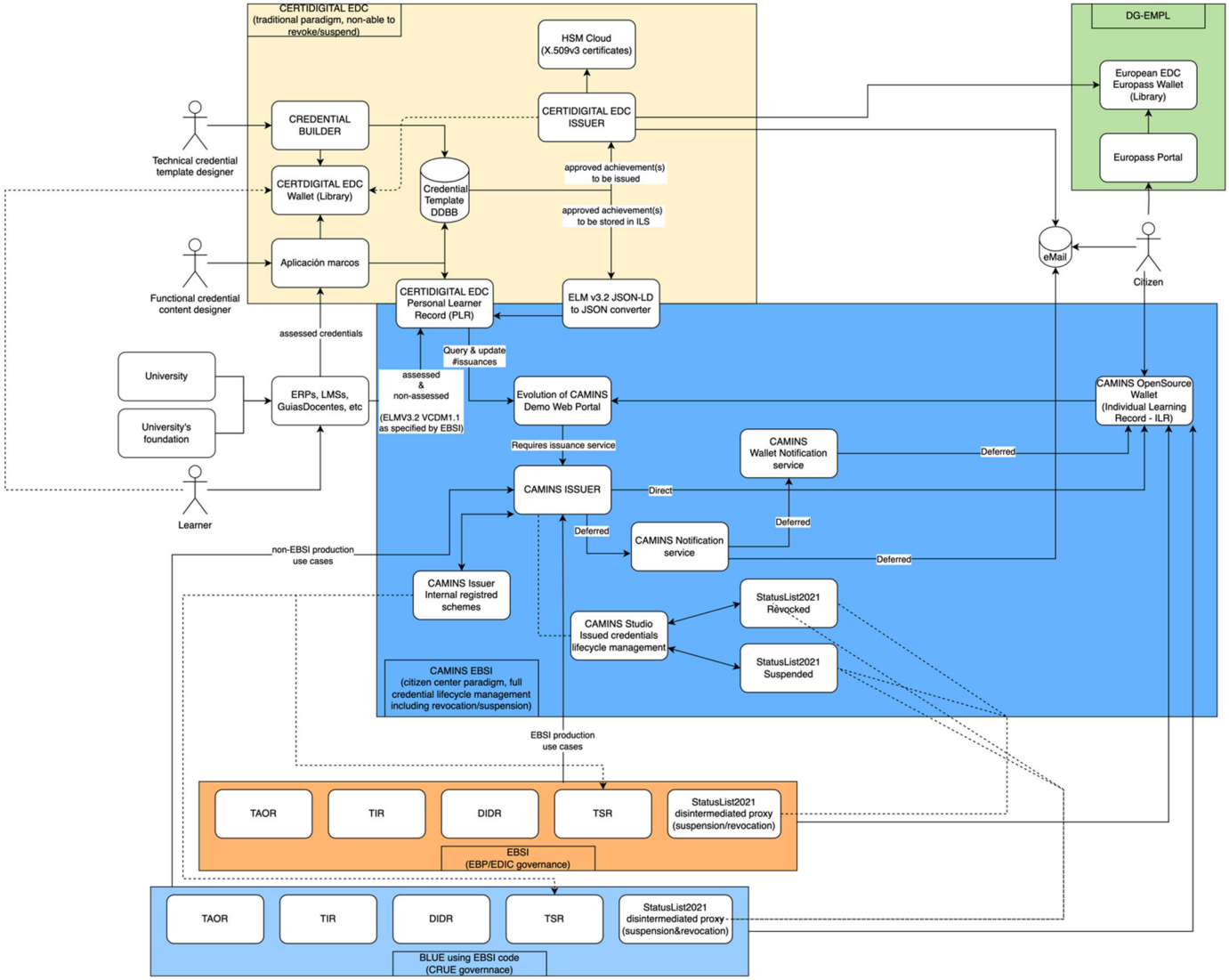
A l'Annex II presentem un exemple de codi que es pot adaptar a les necessitats d'aquest projecte

Després l'adjudicació, caldrà fer un primer procés de consultoria per tancar el conjunt de funcionalitats finals i la millor opció per assolir els requisits de negoci exposats, així com fer els desenvolupaments tècnics adaptats a les darreres versions de les APIs i connectors tant d'OpenLMS com de CAMINS EBSI.



Integració amb EDC Issuer

El nou Generador i emissor de sol·licituds de certificacions europees de certificacions ha de contemplar també la integració amb el CERTIDIGITAL EDC Issuer de forma que el gestor de cada universitat li pugui sol·licitar emissió de certificats/credencials a la vegada que pugui consultar totes les certificacions/acreditacions que el CERTIDIGITAL EDC Issuer hagi signat i gestionat.



Visió arquitectura a alt nivell per la integració CertiDigitalEDC i CAMINS EBSI

En tot cas, i **per a tots els casos:**

- Modelar en format de micro-credencials les acreditacions segons el model CAMINS (i també per al que es gestioni amb CERTIDIGITAL), seguint les recomanacions del Consell d'Europa, incloent un procés de consens d'atributs finals (obligatoris i aquells que hi hagi consens entre les universitats per als opcionals). Per fer-ho, caldrà utilitzar i JSON per CAMINS EBSI.
- Proporcionar funcionalitats de gestió que permetin a cada universitat:
  - Gestionar l'històric de peticions rebudes i emissions sol·licitades i gestionades, que la vegada controli

emissions ja realitzades o enviades per la seva certificació i per evitar duplicitat de certificacions.

- Consultar les accions realitzades amb diferents filtres i criteris
- Traslladar (exportar) els documents certificats generats a altres entorns (propis de cada universitats).
- Conèixer les "notificacions"/enviaments realitzades des del propi Gestor de certificats als "alumnes" (a demanda i rebent alertes o informes periòdics o en el moment de les emissions i notificacions). Ha de ser configurable.
- Gestió dels segells electrònics a utilitzar per cada universitat i "col·lectiu" o "perfil objecte de les emissions de certificacions per l'opció PDF/A. Aquest poden estar allotjats en les respectives universitats o en altres organismes o en la plataforma de custodia al núvol que ofereix el CSUC (PCCD)

- Proporcionar un portal de consulta i visualització de la credencial generada.
- Ha de ser Multi tenen (i que cada Universitat pugui tenir comportaments i usos diferents).
- Qualsevol integració i connexió amb tercers haurà de ser parametritzable (objecte: poder extreure tot i funciona de forma autònoma a cada universitat i amb diferents components)

Altres requeriments

El gestor de Certificacions generat serà de format obert i en tot cas propietat de les universitats participants en el projecte. En cap cas de l'empresa adjudicatària.

Es demana també:

- Definició inicial del projecte i aprovació del model per part de tots els participants.
- Definició d'un cas d'ús per cada universitat participant.
- Documentació detallada de tot els mòduls generats, la seva arquitectura, el seu funcionament i ús, així com una guia específica per a integradors.
- Implementació i funcionament de com a mínim un cas d'ús per cada universitat i un mínim de 10 certificacions per universitat (emitides, enviades/notificades, verificació i visualització realitzada i evidència reportada a la universitat.
- Formació (per a les 6 universitats) per a perfils tècnics i per a usuaris)
- Garantia
- Suport
-

Barcelona,

Vicegerent de Transformació Digital, Dades i TIC

Universitat de Barcelona

Annex I – (Model 1) **Visió interacció OpenLMS i generació PDF/A**

Passos

1. Autenticació i Obtenció de Token: Autenticar-se amb OpenLMS i RedTrust i obtenir els tokens JWT necessaris.
2. Recuperació dels Estudiants Completats: Descarregar els logros educatius dels estudiants des d'OpenLMS.
3. Filtratge dels Estudiants per Universitats: Identificar els estudiants que han completat els cursos i agrupar-los per universitat.
4. Generació de Certificats PDF/A: Generar els certificats en format PDF/A utilitzant les plantilles definides per cada universitat.
5. Signatura Digital dels Certificats: Signar digitalment els certificats utilitzant els certificats dipositats en RedTrust.
6. Enviament de Certificats per Email: Enviar els certificats signats per correu electrònic als estudiants.
7. Gestió de Logs i Històrics: Mantenir logs detallats de totes les operacions i conservar els històrics de les certificacions emeses.

Exemple Il·lustratiu

Aquest codi il·lustra el procés complet des de l'autenticació fins a l'enviament dels certificats signats per correu electrònic. Cada secció de codi inclou comentaris detallats per a la seva comprensió i adaptació a necessitats específiques.

```
import requests
import json
import schedule
import time
from fpdf import FPDF
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.application import MIMEApplication
from email.mime.text import MIMEText
import os

# Configuració de les variables
openlms_api_url = 'https://openlms.example.com/api'
openlms_api_key = 'your_openlms_api_key'
redtrust_api_url = 'https://redtrust.example.com/api'
redtrust_api_key = 'your_redtrust_api_key'
smtp_server = 'smtp.example.com'
smtp_port = 587
smtp_user = 'your_smtp_user'
smtp_password = 'your_smtp_password'
from_email = 'no-reply@example.com'

# Funció per autenticar-se amb OpenLMS i obtenir un token JWT
```

```
def autenticar_openlms(api_url, api_key):
    headers = {
        'Authorization': f'Bearer {api_key}',
        'Content-Type': 'application/json'
    }
    response = requests.get(f'{api_url}/authenticate', headers=headers)
    return response.json()['access_token']

# Funció per autenticar-se amb RedTrust i obtenir un token JWT
def autenticar_redtrust(api_url, api_key):
    headers = {
        'Authorization': f'Bearer {api_key}',
        'Content-Type': 'application/json'
    }
    response = requests.get(f'{api_url}/authenticate', headers=headers)
    return response.json()['access_token']

# Funció per obtenir el llistat de cursos des d'OpenLMS
def obtenir_llistat_cursos(api_url, api_key):
    headers = {
        'Authorization': f'Bearer {api_key}',
        'Content-Type': 'application/json'
    }
    response = requests.get(f'{api_url}/cursos', headers=headers)
    return response.json()

# Funció per obtenir els estudiants d'un curs des d'OpenLMS
def obtenir_estudiants_per_curs(api_url, api_key, curs_id):
    headers = {
        'Authorization': f'Bearer {api_key}',
        'Content-Type': 'application/json'
    }
    response = requests.get(f'{api_url}/cursos/{curs_id}/estudiants', headers=headers)
    return response.json()

# Funció per obtenir la plantilla del certificat per una universitat i un curs
def obtenir_plantilla_curs(universitat, curs):
    plantilla_path = f'plantilles/{universitat}/{curs}.pdf'
    if os.path.exists(plantilla_path):
        return plantilla_path
    else:
        return 'plantilles/default.pdf'

# Funció per generar un certificat en format PDF/A utilitzant una plantilla
def generar_certificat_pdf(estudiant, curs, universitat):
    plantilla_path = obtenir_plantilla_curs(universitat, curs)
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt=f"Certificat de Compleció del Curs: {curs}", ln=True, align='C')
```

```
    pdf.cell(200, 10, txt=f"Nom: {estudiant['nom']}", ln=True, align='L')
    pdf.cell(200, 10, txt=f"Cognoms: {estudiant['cognoms']}", ln=True, align='L')
    pdf.cell(200, 10, txt=f"Data de Compleció: {estudiant['data_complecio']}", ln=True, align='L')
    pdf_output = f"certificats/{universitat}/{estudiant['id']}.pdf"
    pdf.output(pdf_output, 'F')
    return pdf_output

# Funció per signar digitalment un certificat utilitzant RedTrust
def signar_certificat_redtrust(api_url, api_key, universitat, certificat_path):
    headers = {
        'Authorization': f'Bearer {api_key}',
        'Content-Type': 'application/json'
    }
    files = {'file': open(certificat_path, 'rb')}
    response = requests.post(f'{api_url}/sign/{universitat}', headers=headers, files=files)
    signed_certificat_path = certificat_path.replace('.pdf', '_signed.pdf')
    with open(signed_certificat_path, 'wb') as f:
        f.write(response.content)
    return signed_certificat_path

# Funció per enviar un correu electrònic amb el certificat adjunt
def enviar_email(destinatari, assumpte, missatge, fitxer_adjunt):
    msg = MIMEMultipart()
    msg['From'] = from_email
    msg['To'] = destinatari
    msg['Subject'] = assumpte

    body = MIMEText(missatge, 'plain')
    msg.attach(body)

    with open(fitxer_adjunt, 'rb') as attachment:
        part = MIMEApplication(attachment.read(), _subtype="pdf")
        part.add_header('Content-Disposition', f'attachment; filename="{fitxer_adjunt}"')
        msg.attach(part)

    server = smtplib.SMTP(smtp_server, smtp_port)
    server.starttls()
    server.login(smtp_user, smtp_password)
    text = msg.as_string()
    server.sendmail(from_email, destinatari, text)
    server.quit()

# Funció principal que executa tot el procés
def processar_certificacions():
    openlms_token = autenticar_openlms(openlms_api_url, openlms_api_key)
    redtrust_token = autenticar_redtrust(redtrust_api_url, redtrust_api_key)
    llistat_cursos = obtenir_llistat_cursos(openlms_api_url, openlms_token)

    for curs in llistat_cursos:
        estudiants = obtenir_estudiants_per_curs(openlms_api_url, openlms_token, curs['id'])
```

```
        for estudiant in estudiants:
            universitat = estudiant['institucio']
            certificat_path = generar_certificat_pdf(estudiant, curs['nom'], universitat)
            certificat_signat_path = signar_certificat_redtrust(redtrust_api_url, redtrust_token, universitat,
certificat_path)
            enviar_email(estudiant['email'], 'Certificat de Compleció', 'Adjunt trobaràs el teu certificat de compleció del
curs.', certificat_signat_path)

# Programar la periodicitat del procés
schedule.every().day.at("00:00").do(processar_certificacions)

# Mantenir el programa en execució per a la programació
while True:
    schedule.run_pending()
    time.sleep(1)
```

Annex II – (Model 2) **Visió interacció OpenLMS i CAMINS EBSI**

Exemple Il·lustratiu

En aquest punt, es proporciona un exemple de codi que il·lustra el procés complet des de l'autenticació fins a la notificació al wallet de l'usuari. Cada secció de codi inclou comentaris detallats per a la seva comprensió i adaptació a necessitats específiques.

L'exemple de codi proporcionat es pot adaptar a les necessitats específiques dels usuaris i configurar-se per a executar-se de manera periòdica, assegurant la privacitat i la seguretat de les dades de cada usuari.

```
import requests
import json

# Configuració de les variables
ebsi_auth_api_url = 'https://auth.ebsi.example.com'
ebsi_studio_api_url = 'https://studio.ebsi.example.com'
client_id = 'your_client_id'
client_secret = 'your_client_secret'
username = 'your_username'
password = 'your_password'
wallet_callback_url = 'https://wallet.example.com/callback'

# Funció per autenticar-se amb EBSI i obtenir un token JWT
def autenticar_ebsi(auth_api_url, client_id, client_secret, username, password):
    token_url = f'{auth_api_url}/realms/ebsi/protocol/openid-connect/token'
    payload = {
        'client_id': client_id,
        'client_secret': client_secret,
        'username': username,
        'password': password,
        'grant_type': 'password',
        'scope': 'openid'
    }
    response = requests.post(token_url, data=payload)
    return response.json()['access_token']

# Funció per generar una petició de credencial verificable "EducationalID"
def generar_peticio_vc(api_url, token):
    headers = {
        'Authorization': f'Bearer {token}',
        'Content-Type': 'application/json'
    }
    data = {
        "issuanceId": "random-issuance-id",
        "credentialTypeId": "educational_id",
        "credentialSubject": {
```

```
            "id": "did:example:123",
            "name": "John Doe",
            "birthDate": "1990-01-01"
        },
        "issuerDid": "did:web:issuer.example.com",
        "callbackUrl": wallet_callback_url
    }
    response = requests.post(f'{api_url}/camins-studio/api/v1/credential-offer-pre-auth', headers=headers, json=data)
    return response.json()

# Funció per crear una presentació verificable (VP)
def crear_vp(api_url, token, educational_id_vc):
    headers = {
        'Authorization': f'Bearer {token}',
        'Content-Type': 'application/json'
    }
    data = {
        "scope": "EducationalID",
        "verificationRequestId": "random-verification-request-id",
        "callbackUrl": wallet_callback_url
    }
    response = requests.post(f'{api_url}/camins-studio/api/v1/oidc4vp/vp-request', headers=headers, json=data)
    return response.json()

# Funció per validar una presentació verificable (VP)
def validar_vp(api_url, token, vp):
    headers = {
        'Authorization': f'Bearer {token}',
        'Content-Type': 'application/json'
    }
    data = {
        "vp": vp
    }
    response = requests.post(f'{api_url}/camins-studio/api/v1/validate-vp', headers=headers, json=data)
    return response.json()

# Funció per recuperar els logros acadèmics basats en l'atribut "ABC"
def recuperar_logros(api_url, token, atribut_abc):
    headers = {
        'Authorization': f'Bearer {token}',
        'Content-Type': 'application/json'
    }
    response = requests.get(f'{api_url}/logros?atribut={atribut_abc}', headers=headers)
    return response.json()

# Funció per generar les acreditacions seleccionades
def generar_acreditacions(api_url, token, logros_seleccionats, user_did):
    headers = {
        'Authorization': f'Bearer {token}',
        'Content-Type': 'application/json'
```

```
    }
    for logro in logros_seleccionats:
        data = {
            "issuanceId": "random-issuance-id",
            "entityDid": "did:web:entity.example.com",
            "credentialTypeId": "microcredential",
            "recipientDid": user_did,
            "payload": {
                "logro": logro
            }
        }
        response = requests.post(f'{api_url}/camins-studio/api/v1/credentials', headers=headers, json=data)
        print(response.json())

# Funció principal que executa tot el procés
def processar_acreditacions():
    token = autenticar_ebsi(ebsi_auth_api_url, client_id, client_secret, username, password)
    educational_id_vc = generar_peticio_vc(ebsi_studio_api_url, token)
    vp = crear_vp(ebsi_studio_api_url, token, educational_id_vc)
    vp_validada = validar_vp(ebsi_studio_api_url, token, vp)
    if vp_validada['status'] == 'valid':
        atribut_abc = vp_validada['credentialSubject']['ABC']
        logros = recuperar_logros(ebsi_studio_api_url, token, atribut_abc)
        logros_seleccionats = []  # Seleccionar logros manualment per a aquest exemple
        user_did = "did:example:123"
        generar_acreditacions(ebsi_studio_api_url, token, logros_seleccionats, user_did)

# Programar la periodicitat del procés
schedule.every().day.at("00:00").do(processar_acreditacions)

# Mantenir el programa en execució per a la programació
while True:
    schedule.run_pending()
    time.sleep(1)
```