🧩 SKILL SPROUT

Microcertificaciones verificables de habilidades específicas



# Descripción

SkillSprout es una app móvil que permite a los usuarios recolectar microcertificaciones verificables en tiempo real dentro de una wallet personal. Cada logro concreto en plataformas educativas, hackathons, comunidades tecnológicas o bootcamps se convierte en un microbadge on-chain que respalda la adquisición de una habilidad puntual (por ejemplo, "Resolución de 5 problemas de complejidad algorítmica media en Python", o "Primer lugar en reto UX/UI").

Construida con Starknet.dart y Flutter, la app permite visualizar el progreso de aprendizaje en una interfaz gamificada y social. Este enfoque fomenta la motivación continua, facilita la construcción de un portafolio confiable y detallado, y ofrece trazabilidad transparente para reclutadores, instituciones educativas y comunidades tecnológicas.

Contents

[Descripción 1](#_Toc198582559)

[Diferenciador 3](#_Toc198582560)

[🧠 ¿Qué es? 3](#_Toc198582561)

[🛠️ Tecnologías: 3](#_Toc198582562)

[🔥 ¿Por qué destaca? 3](#_Toc198582563)

[🧭 Sector 3](#_Toc198582564)

[✅ Categorías (Tags) 4](#_Toc198582565)

[Idea de Código Inicial 4](#_Toc198582566)

[1. Planeación del Proyecto 4](#_Toc198582567)

[2. Backend 4](#_Toc198582568)

[3. Contrato en Cairo (Starknet) 4](#_Toc198582569)

[4. Flutter + starknet.dart 5](#_Toc198582570)

[5. Lógica de microcertificaciones 6](#_Toc198582571)

[6. Detalles técnicos para destacar en el jurado 6](#_Toc198582572)

[7. Demo final y presentación 7](#_Toc198582573)

[8. 📦 Repositorio 7](#_Toc198582574)

[Estructura de Carpetas Flutter 7](#_Toc198582575)

[📦 pubspec.yaml (fragmento clave) 8](#_Toc198582576)

[📜 Contrato Cairo: skillchain.cairo 9](#_Toc198582577)

[Código del main.dart / get\_badge e issue\_badge y Demo Interactivo Básico 9](#_Toc198582578)

[🧠 1. main.dart – Punto de entrada básico 9](#_Toc198582579)

[🏠 2. screens/home\_screen.dart 10](#_Toc198582580)

[🔌 3. services/starknet\_service.dart 12](#_Toc198582581)

[🧩 4. widgets/badge\_card.dart 13](#_Toc198582582)

# Diferenciador

SkillSprout redefine cómo se validan y muestran las habilidades técnicas y creativas. A diferencia de los certificados tradicionales de cursos completos, cada microhabilidad adquirida —ya sea completar ejercicios, ganar retos o colaborar en proyectos— se convierte en una microcertificación única, emitida como un badge verificable en la blockchain.

En lugar de emitir certificados genéricos de cursos, cada habilidad o logro individual se certifica (ej: "Completó 3 ejercicios de algoritmos en Python", "Ganó reto de diseño UX"). Microbadges en la blockchain.

# 🧠 ¿Qué es?

Una app donde los usuarios acumulan microcertificaciones en su wallet por logros concretos en plataformas educativas, hackathons, comunidades, etc.

Pueden formar parte de un portafolio verificado de habilidades (como "insignias" on-chain).

# 🛠️ Tecnologías:

* Pragma: contrato inteligente de certificación por microhabilidad.
* Starknet.dart + Flutter: mostrar badges, logros y progreso en una app móvil.

# 🔥 ¿Por qué destaca?

* Permite gamificación del aprendizaje.
* Muestra crecimiento progresivo en tiempo real.
* Aplica para bootcamps, plataformas de retos, clubes de programación.

# 🧭 Sector

* EdTech (Tecnología Educativa)
* Blockchain y Web3
* Gamificación del Aprendizaje
* Portafolios Profesionales y Verificación de Skills
* Comunidades y Bootcamps de Programación

# ✅ Categorías (Tags)

* SocialFi – SkillSprout permite a los usuarios construir una reputación social basada en logros verificables, lo cual puede integrarse con perfiles sociales y redes profesionales Web3.
* NFT – Cada microcertificación o "badge" funciona como un NFT único y verificable on-chain.
* Infra – Utiliza infraestructura blockchain (Pragma + Starknet) para contratos inteligentes de certificación.
* AI (integración futura) – Modelos de AI para evaluar ejercicios, recomendar próximos retos o detectar progreso en el aprendizaje.

# Idea de Código Inicial

## Planeación del Proyecto

🎯 Objetivo: Crear una app móvil donde los usuarios puedan recibir y visualizar microcertificaciones ("badges") por logros educativos, guardadas on-chain en Starknet.

## Backend

* ✅ Recibe eventos o datos de plataformas educativas (ej. “completó reto”)
* ✅ Valida y llama al contrato en Starknet para emitir un badge
* Herramientas: Node.js, Flask, FastAPI, etc.

## Contrato en Cairo (Starknet)

* 🧱 Objetivo: Emitir microcertificaciones on-chain como registros únicos vinculados a wallets.
* 📝 Estructura básica:
* Cairo

%lang starknet

@storage\_var

func badge\_by\_user(user: felt, badge\_id: felt) -> (description: felt):

end

@external

func issue\_badge(user: felt, badge\_id: felt, description: felt):

badge\_by\_user.write(user, badge\_id, description)

return ()

end

@view

func get\_badge(user: felt, badge\_id: felt) -> (description: felt):

let (desc) = badge\_by\_user.read(user, badge\_id)

return (description=desc)

end

* 🚀 Compilar y desplegar
* Starkli o starknet CLI:

starkli compile skillchain.cairo

starkli declare ...

starkli deploy ...

## Flutter + starknet.dart

flutter create skillchain\_app

cd skillchain\_app

* 🧩 Agregar dependencias

yaml

dependencies:

lutter:

sdk: flutter

starknet: ^x.y.z # versión más reciente

* 🔌 Conexión con Starknet

Conectar con wallet (ArgentX / Braavos)

Llamar a get\_badge() para consultar logros

Llamar a issue\_badge() (backend o admin)

* 📱 Interfaz Sugerida

|  |  |
| --- | --- |
| Pantalla | Funcionalidad |
| Home | Lista de insignias (badges) del usuario |
| Badge Detail | Detalles: descripción, fecha, ID |
| Conectar Wallet | Autenticación |
| Emitir Badge | (solo para admin/backend) |

* Ejemplo: Llamar un contrato

final provider = JsonRpcProvider(nodeUri: 'https://...');

final contract = StarknetContract(

address: '0x...', // dirección del contrato

abi: [...], // ABI del contrato en formato Dart

provider: provider,

);

// Llamar view

final result = await contract.call('get\_badge', [userAddress, badgeId]);

## Lógica de microcertificaciones

* Cada badge debe tener:
* ✅ ID único
* 🧾 Descripción (ej. "Completó 3 retos en Python")
* 📅 Fecha (puede ser off-chain si no cabe on-chain)
* 🧠 Fuente (bootcamp, hackathon, etc.)

## Detalles técnicos para destacar en el jurado

🟢 Best Use of Starknet

* ✅ Muestras lectura y escritura de datos on-chain reales
* ✅ Estructura eficiente (1 usuario puede tener múltiples badges)
* ✅ Diseño modular: puede integrarse a otros ecosistemas

🟣 Build Mobile Apps with starknet.dart

* ✅ UI atractiva, gamificada
* ✅ Wallet integrada
* ✅ Carga dinámica de logros por wallet

## Demo final y presentación

* Conexión con wallet
* Consulta de badges
* Recepción de un nuevo badge
* Cómo se ven en la app móvil

## 📦 Repositorio

* Código del contrato Cairo
* Código Flutter
* README explicando el flujo y tecnologías

# Estructura de Carpetas Flutter

skillchain\_app/

├── lib/

│ ├── main.dart

│ ├── screens/

│ │ ├── home\_screen.dart

│ │ ├── badge\_detail\_screen.dart

│ │ └── wallet\_connect\_screen.dart

│ ├── widgets/

│ │ └── badge\_card.dart

│ └── services/

│ └── starknet\_service.dart

├── assets/

├── pubspec.yaml

# 📦 pubspec.yaml (fragmento clave)

name: skillchain\_app

description: Microcertificaciones verificables on-chain con Starknet y Flutter.

version: 1.0.0+1

environment:

sdk: '>=3.0.0 <4.0.0'

dependencies:

flutter:

sdk: flutter

starknet: ^0.4.1 # Verifica la versión más reciente en pub.dev

cupertino\_icons: ^1.0.2

provider: ^6.0.0

http: ^0.14.0

dev\_dependencies:

flutter\_test:

sdk: flutter

# 📜 Contrato Cairo: skillchain.cairo

%lang starknet

@storage\_var

func badge\_by\_user(user: felt, badge\_id: felt) -> (description: felt):

end

@external

func issue\_badge(user: felt, badge\_id: felt, description: felt):

badge\_by\_user.write(user, badge\_id, description)

return ()

end

@view

func get\_badge(user: felt, badge\_id: felt) -> (description: felt):

let (desc) = badge\_by\_user.read(user, badge\_id)

return (description=desc)

end

# Código del main.dart / get\_badge e issue\_badge y Demo Interactivo Básico

## 🧠 1. main.dart – Punto de entrada básico

import 'package:flutter/material.dart';

import 'screens/home\_screen.dart';

void main() {

runApp(const SkillChainApp());

}

class SkillChainApp extends StatelessWidget {

const SkillChainApp({super.key});

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'SkillChain',

theme: ThemeData(

primarySwatch: Colors.indigo,

),

home: const HomeScreen(),

);

}

}

## 🏠 2. screens/home\_screen.dart

import 'package:flutter/material.dart';

import '../services/starknet\_service.dart';

import '../widgets/badge\_card.dart';

class HomeScreen extends StatefulWidget {

const HomeScreen({super.key});

@override

State<HomeScreen> createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

List<Map<String, dynamic>> badges = [];

final String userWallet = '0x123...'; // Wallet real del usuario

@override

void initState() {

super.initState();

loadBadges();

}

Future<void> loadBadges() async {

final loaded = await StarknetService.getBadges(userWallet);

setState(() {

badges = loaded;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('SkillChain Badges'),

),

body: ListView.builder(

itemCount: badges.length,

itemBuilder: (context, index) {

return BadgeCard(badge: badges[index]);

},

),

);

}

}

## 🔌 3. services/starknet\_service.dart

import 'package:starknet/starknet.dart';

class StarknetService {

static const String contractAddress = '0x...'; // Dirección de tu contrato

static const String nodeUrl = 'https://alpha4.starknet.io/rpc'; // Testnet

static final provider = JsonRpcProvider(nodeUri: nodeUrl);

static final abi = [ /\* ABI aquí \*/ ];

static final contract = StarknetContract(

address: contractAddress,

abi: abi,

provider: provider,

);

static Future<List<Map<String, dynamic>>> getBadges(String user) async {

List<Map<String, dynamic>> badges = [];

for (int i = 0; i < 5; i++) {

try {

final result = await contract.call(

'get\_badge',

[Felt.fromHexString(user), Felt.fromInt(i)],

);

final desc = String.fromCharCodes(result[0].value.toBigInt().toRadixString(16).codeUnits);

if (desc.isNotEmpty) {

badges.add({'id': i, 'description': desc});

}

} catch (\_) {}

}

return badges;

}

}

## 🧩 4. widgets/badge\_card.dart

import 'package:flutter/material.dart';

class BadgeCard extends StatelessWidget {

final Map<String, dynamic> badge;

const BadgeCard({super.key, required this.badge});

@override

Widget build(BuildContext context) {

return Card(

margin: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),

child: ListTile(

leading: const Icon(Icons.verified, color: Colors.green),

title: Text('Badge #${badge['id']}'),

subtitle: Text(badge['description']),

),

);

}

}