

# DESCLASIFICACIÓN BASADA EN TIPOS EN DART

## IMPLEMENTACIÓN Y ELABORACIÓN DE HERRAMIENTAS DE INFERENCIA

---

Matías Meneses Cortés

1. Control de flujo de información
2. Inferencia de tipos
3. Inferencia de facetas públicas en Dart
4. Validación
5. Conclusión

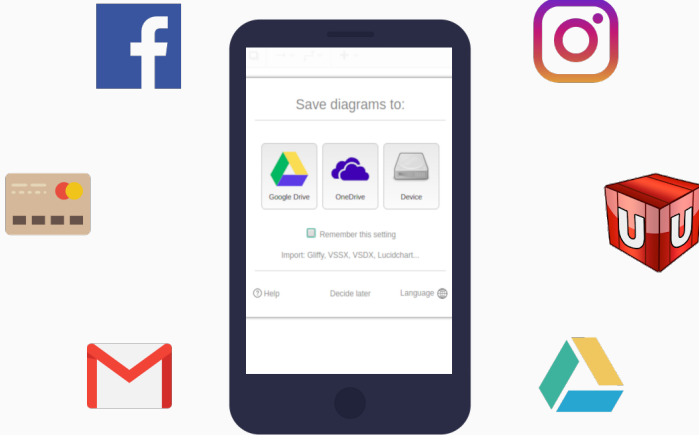
# Control de flujo de información

---

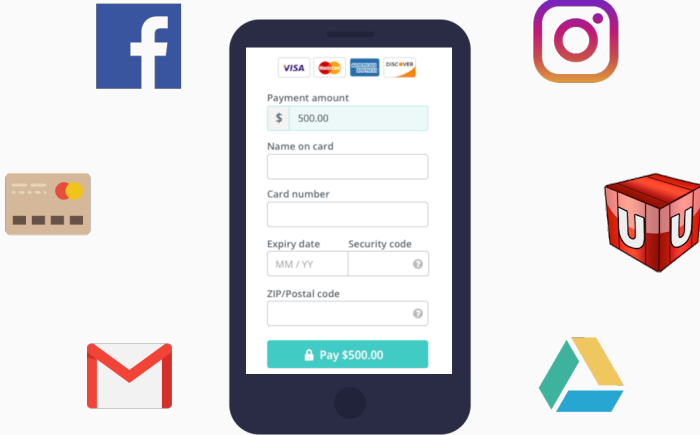


# Protección de confidencialidad





# Protección de confidencialidad



# Protección de confidencialidad

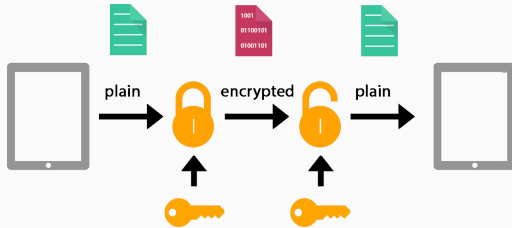




Distintas técnicas de seguridad en distintas capas de comunicación.

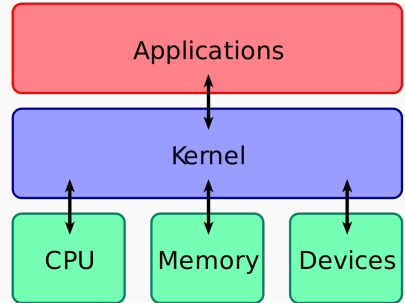
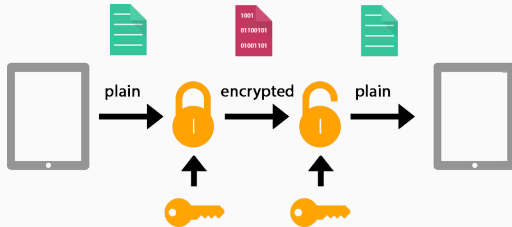
# Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.



# Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.



# Seguridad basada en el lenguaje: Tipado de seguridad

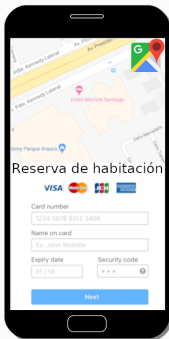


```
String@L login(String@L guess, String@H password) {  
    if (password == guess) return "Login successful";  
    else return "Login failed";  
}
```

H  
|  
L

Orden parcial de dos niveles

# Control de flujo de información



```
String book(String username, int date, int cardNumber) {  
    return sendToHotel(username, date, cardNumber);  
}
```

```
String sendToHotel(String username, int date, int cardNumber);  
String sendToGoogle(String token, int xCoord, int yCoord);
```

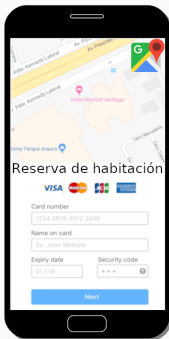
# Control de flujo de información



```
String book(String username, int date, int cardNumber) {  
    return sendToGoogle (username, date, cardNumber);  
}
```

```
String sendToHotel(String username, int date, int cardNumber);  
String sendToGoogle(String token, int xCoord, int yCoord);
```

# Tipado de seguridad para el control de flujo de información



```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToGoogle(username, date, cardNumber);  
}
```

```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```

# Tipado de seguridad para el control de flujo de información

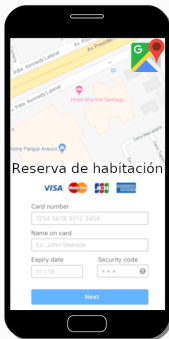


```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToGoogle(username, date, cardNumber);  
}
```

```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```



# Tipado de seguridad para el control de flujo de información

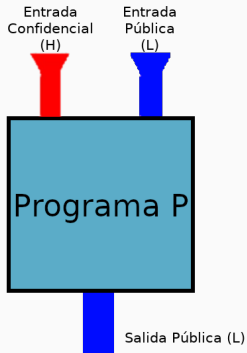


```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToHotel(username, date, cardNumber);  
}
```

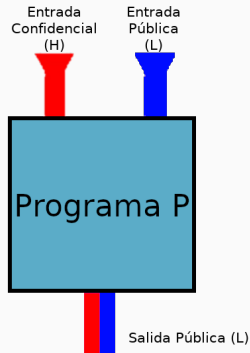
```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```

# No-interferencia

Propiedad fundamental del control de flujo de información.



No-interferencia



Fuga de información

```
String@L login(String@L guess, String@H password) {  
    if (password == guess) return "Login successful";  
    else return "Login failed";  
}
```

¡No cumple con no-interferencia!

```
String@L login(String@L guess, String@H password) {  
    if (declassify(password == guess)) return "Login successful";  
    else return "Login failed";  
}
```

```
declassify(password)
```

```
declassify(password)
```

¡Grave fuga de información!

```
String<String login(String<String guess, String<StringEq password) {  
    if (password.eq(guess)) return "Login successful";  
    else return "Login failed";  
}
```

```
String<String login(String<String guess, String<StringEq password) {  
    if (password.eq(guess)) return "Login successful";  
    else return "Login failed";  
}
```

- Tipos de dos facetas `String<StringEq`



## Desclasificación basada en tipos

```
String<String login(String<String guess, String<StringEq password) {  
  if (password.eq(guess)) return "Login successful";  
  else return "Login failed";  
}
```

- Tipos de dos facetas `String<StringEq`
- `StringEq = [eq: String<String -> Bool<Bool]`

## Desclasificación basada en tipos

```
String<String login(String<String guess, String<StringEq password) {  
  if (password.eq(guess)) return "Login successful";  
  else return "Login failed";  
}
```

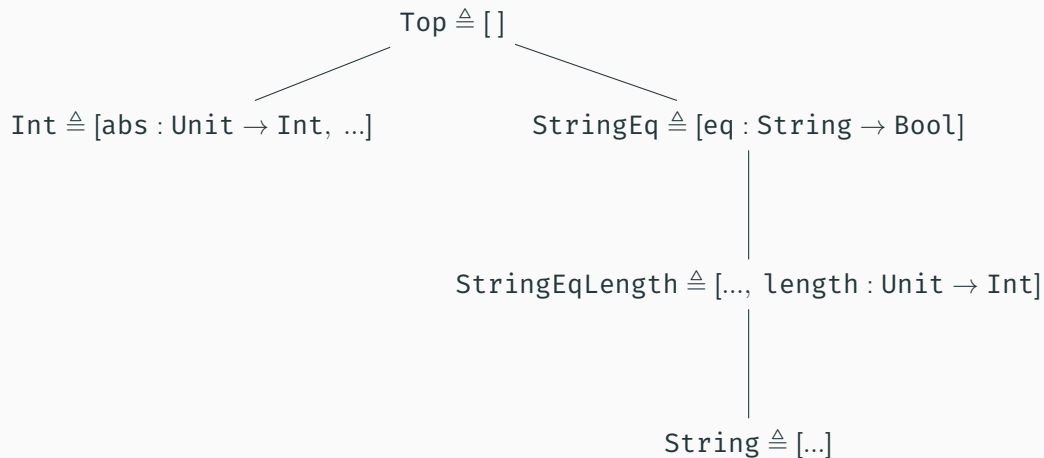
- Tipos de dos facetas `String<StringEq`
- `StringEq = [eq: String<String -> Bool<Bool]`
- `String <: StringEq` (Tipo bien formado)

## Desclasificación basada en tipos

```
String<String login(String<String guess, String<StringEq password) {  
  if (password.eq(guess)) return "Login successful";  
  else return "Login failed";  
}
```

- Tipos de dos facetas `String<StringEq`
- `StringEq = [eq: String<String -> Bool<Bool]`
- `String <: StringEq` (Tipo bien formado)
- No-interferencia relajada

## Retículo de subtipos



## Regla principal de la desclasificación basada en tipos

```
String<String login(String<String guess, String<Top password) {  
    if ( password.eq(guess) ) return "Login successful";  
    else return "Login failed";  
}
```

## Regla principal de la desclasificación basada en tipos

```
String<String login(String<String guess, String<Top password) {  
  if ( password.eq(guess) ) return "Login successful";  
  else return "Login failed";  
}
```

## Problemas con la desclasificación basada en tipos

- Propuesta sin implementación práctica.



## Problemas con la desclasificación basada en tipos

- Propuesta sin implementación práctica.
- Anotación completa de facetas para realizar análisis.

# Problemas con la desclasificación basada en tipos

- Propuesta sin implementación práctica.
- Anotación completa de facetas para realizar análisis.

```
String login(String guess, String<StringEq password) {  
    if (password.eq(guess)) return "Login successful";  
    else return "Login failed";  
}
```

# Inferencia de tipos

---

```
def mathFunction(n1: Int, n2: Float) = {  
  val n = fact(n1);  
  n + n2;  
}
```

```
def fact(n: Int) : Int = {  
  if (n == 0) return 1  
  else n * fact(n-1)  
}
```

```
def mathFunction(n1: Int, n2: Float) = {  
  val n = fact(n1);  
  n + n2;  
}
```

```
def fact(n: Int) : Int = {  
  if (n == 0) return 1  
  else n * fact(n-1)  
}
```

- Inferencia de tipos local

```
def mathFunction(n1: Int, n2: Float) = {  
  val n = fact(n1);  
  n + n2;  
}
```

```
def fact(n: Int) : Int = {  
  if (n == 0) return 1  
  else n * fact(n-1)  
}
```

- Inferencia de tipos local
- Soporte de overloading y conversiones implícitas de tipos

```
# let average a b =  
    (a +. b) /. 2.0;;  
  
val average : float -> float -> float
```

```
# let average a b =  
  (a +. b) /. 2.0;;  
  
val average : float -> float -> float
```

- Inferencia de tipos global



```
# let average a b =  
    (a +. b) /. 2.0;;  
  
val average : float -> float -> float
```

- Inferencia de tipos global
- No soporta overloading y conversiones implícitas

## Objetivo de la memoria

Implementar un sistema de inferencia de facetas públicas para la desclasificación basada en tipos, en conjunto con una extensión para ambientes de desarrollo.

## Inferencia de facetas públicas en Dart

---

## Ejemplo 1

```
bool login(String guess, String password) {  
    return password.eq(guess);  
}
```

## Ejemplo 1

```
bool login(String guess, String<StringEq password) {  
    return password.eq(guess);  
}
```

```
StringEq = [eq: String<String -> bool<bool]
```

## Ejemplo 1

```
bool login(String<String guess, String<StringEq password) {  
    return password.eq(guess);  
}
```

## Ejemplo 1

```
bool<bool login(String<String guess, String<StringEq password) {  
    return password.eq(guess);  
}
```

## Ejemplo 2

```
bool login(String<String guess, String<Top password) {  
    return password.eq(guess);  
}
```



## Ejemplo 2

```
bool<Top login(String<String guess, String<Top password) {  
    return password.eq(guess);  
}
```

## Ejemplo 3

```
bool login(int guess, String password) {  
    return password.hash().eq(guess);  
}
```

## Ejemplo 3

```
bool<bool login(String<String guess, String<StringHash password) {  
    return password.hash().eq(guess);  
}
```

```
StringHash = [hash: () -> int< -]
```

## Ejemplo 3

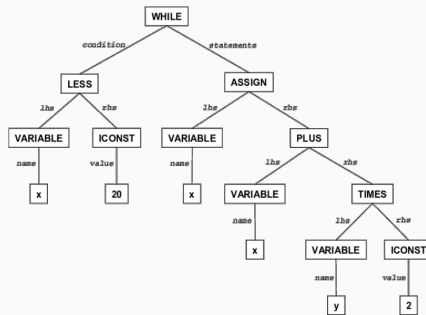
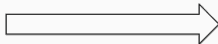
```
bool<bool login(String<String guess, String<StringHash password) {  
    return password.hash().eq(guess);  
}
```

```
StringHash = [hash: () -> int<IntEq]
```

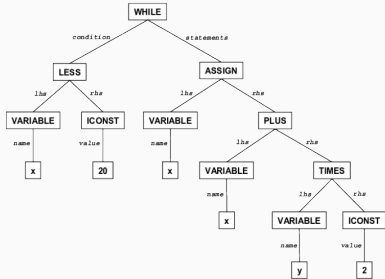
```
IntEq = [eq: int<int -> bool<bool]
```



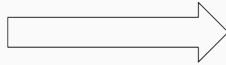
Dart



Abstract Syntax Tree



Abstract Syntax Tree



Errores y sugerencias de edición

```
class Foo {
```



```
class Foo {  
  String foo(String a, String b) {
```

```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
  }  
}
```

```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
    if (a == b)
```

```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
    if (a == b) return a.concat(b);  
    return s;  
  }  
}
```

Uso de anotaciones de Dart para declarar las facetas públicas.

Uso de anotaciones de Dart para declarar las facetas públicas.

```
bool login(String guess, @S("StringEq") String password) {  
  return password.eq(guess);  
}
```

## Definición de facetas públicas

Uso de clases abstractas de Dart para declarar las facetas públicas.

# Definición de facetas públicas

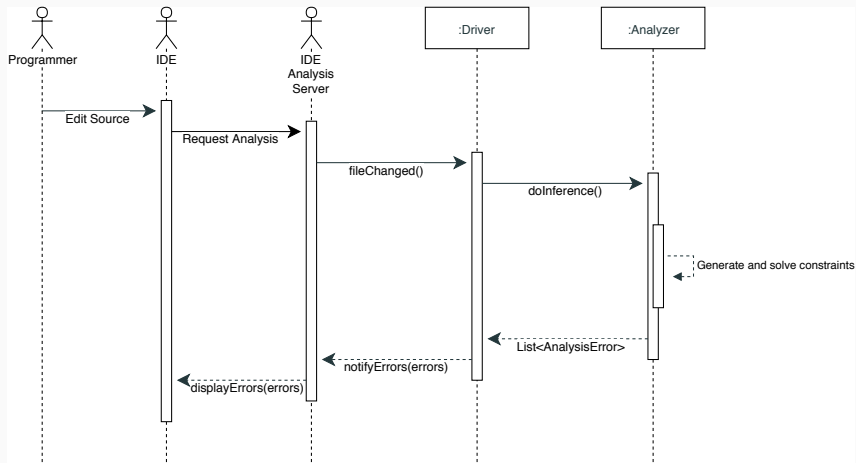
Uso de clases abstractas de Dart para declarar las facetas públicas.

```
bool login(String guess, @S("StringEq") String password) {  
    return password.eq(guess);  
}
```

```
abstract class StringEq {  
    bool eq(String other);  
}
```



# Extensión para ambientes de desarrollo



Errores de seguridad

## Errores de seguridad

```
bool<bool login(String guess, @S("Top") String password) {  
    return password.eq(guess);  
}
```

Errores de tipo mal formado

Errores de tipo mal formado

```
bool<Top login(String guess, @S("Abs") String password) {  
    return password.eq(guess);  
}  
  
abstract class Abs {  
    int abs();  
}
```

Warning de faceta pública no definida

Warning de faceta pública no definida

```
bool<bool login(String guess, @S("StringEq") String password) {  
    return password.eq(guess);  
}
```

Información de faceta pública inferida



Información de faceta pública inferida

```
bool<bool login(String<String guess, String password) {  
    return password.eq(guess);  
}
```

Información de faceta pública inferida

```
bool<bool login(String<String guess, String password) {  
    return password.eq(guess);  
}
```

```
password: [eq: String<String -> bool<bool]
```

# Validación

---

# Ejemplo: Sistema de autenticación web

## Secure Login

Enter your credentials

Enter your email

Enter your password

[Forgot Password?](#)

LOGIN

## Ejemplo: Sistema de autenticación web

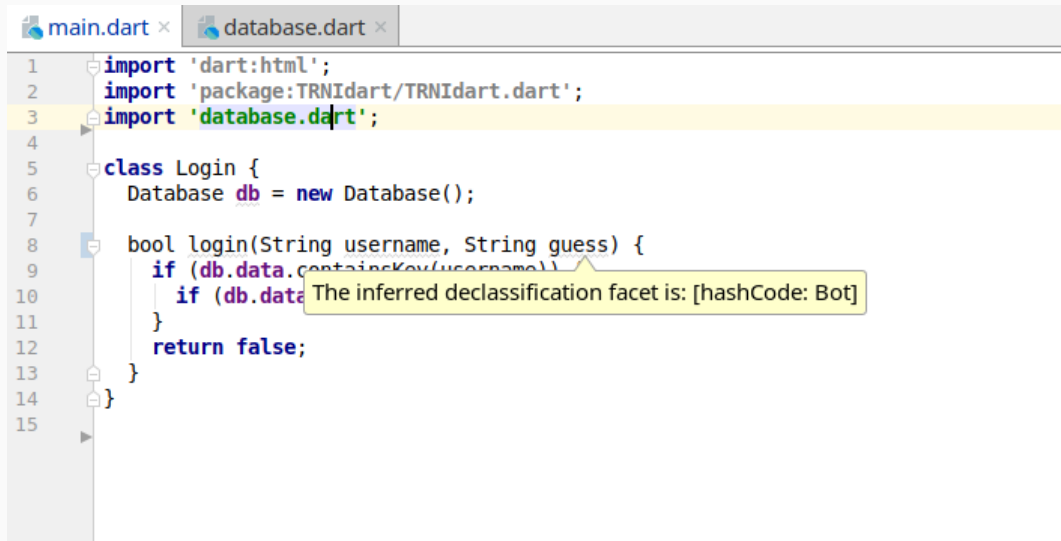
```
main.dart x database.dart x
1 import 'package:TRNIdart/TRNIdart.dart';
2
3
4 abstract class Hash {
5   int get hashCode;
6 }
7
8 abstract class Data {
9   int get data;
10 }
11
12 abstract class ContainsKeyAndGetHash {
13   bool containsKey(Object key);
14   @S("Top") bool operator [](Object key);
15 }
16
17 class Database {
18   @S("Data") Database();
19
20   @S("ContainsKeyAndGetHash") Map<String, int> get data => {
21     "mmeneses@dcc.uchile.cl": "12345".hashCode,
22     "matias.imc@gmail.com": "123456".hashCode,
23   };
24 }
```

## Ejemplo: Sistema de autenticación web



```
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Login {
6   Database db = new Database();
7
8   bool login(String username, String guess) {
9     if (db.data.containsKey(username)) {
10       if (db.data[username] == guess.hashCode) return true;
11     }
12     return false;
13   }
14 }
15
```

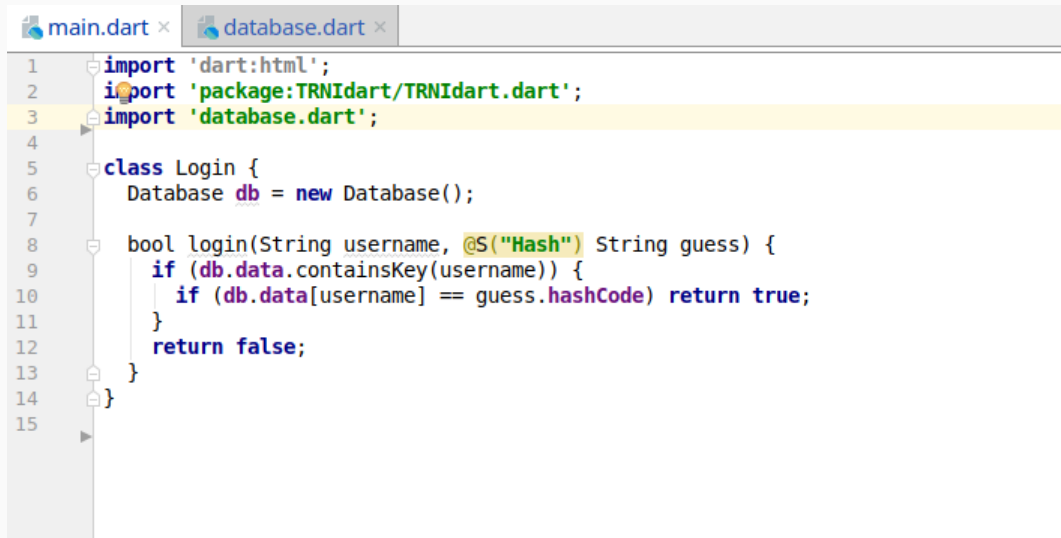
## Ejemplo: Sistema de autenticación web



```
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Login {
6   Database db = new Database();
7
8   bool login(String username, String guess) {
9     if (db.data.containsKey(username)) {
10       if (db.data[username] == guess) {
11         return true;
12       }
13     }
14     return false;
15   }
16 }
```

The inferred declassification facet is: [hashCode: Bot]

## Ejemplo: Sistema de autenticación web



```
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Login {
6   Database db = new Database();
7
8   bool login(String username, @S("Hash") String guess) {
9     if (db.data.containsKey(username)) {
10       if (db.data[username] == guess.hashCode) return true;
11     }
12     return false;
13   }
14 }
15
```



## Ejemplo: Sistema de autenticación web



```
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Login {
6   Database db = new Database();
7
8   bool login(String username, @S("Hash") String guess) {
9     if (db.data.containsKey(username)) {
10       if (db.data[username] == guess) {
11         return true;
12       }
13     }
14     return false;
15   }
16 }
```

The declared facet Hash is not defined.

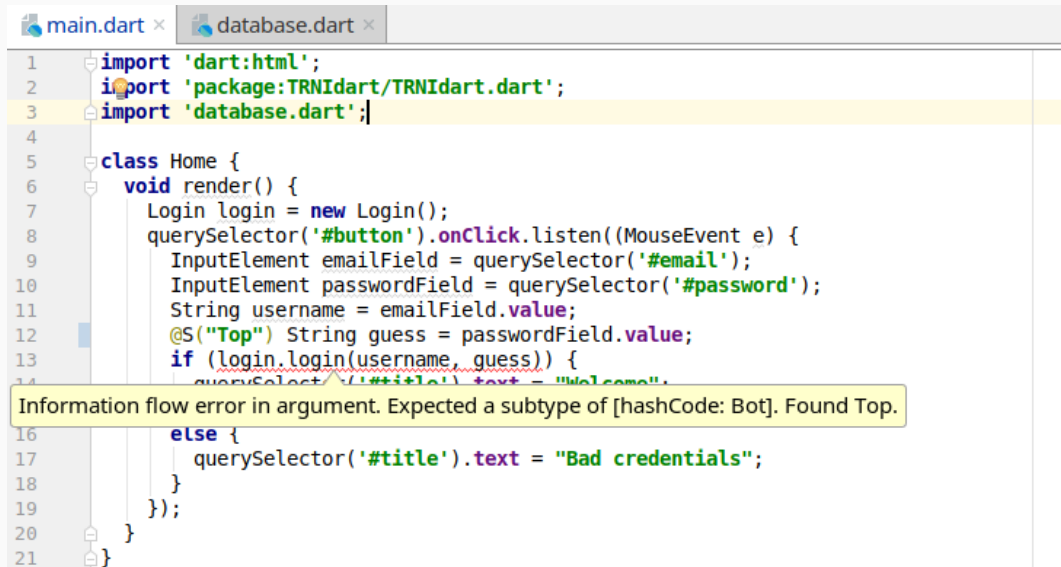
## Ejemplo: Sistema de autenticación web

```
main.dart x database.dart x
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Home {
6   void render() {
7     Login login = new Login();
8     querySelector('#button').onClick.listen((MouseEvent e) {
9       InputElement emailField = querySelector('#email');
10      InputElement passwordField = querySelector('#password');
11      String username = emailField.value;
12      String guess = passwordField.value;
13      if (login.login(username, guess)) {
14        querySelector('#title').text = "Welcome";
15      }
16      else {
17        querySelector('#title').text = "Bad credentials";
18      }
19    });
20  }
21 }
```

## Ejemplo: Sistema de autenticación web

```
main.dart x database.dart x
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Home {
6   void render() {
7     Login login = new Login();
8     querySelector('#button').onClick.listen((MouseEvent e) {
9       InputElement emailField = querySelector('#email');
10      InputElement passwordField = querySelector('#password');
11      String username = emailField.value;
12      @S("Top") String guess = passwordField.value;
13      if (login.login(username, guess)) {
14        querySelector('#title').text = "Welcome";
15      }
16      else {
17        querySelector('#title').text = "Bad credentials";
18      }
19    });
20  }
21 }
```

## Ejemplo: Sistema de autenticación web



```
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Home {
6   void render() {
7     Login login = new Login();
8     querySelector('#button').onClick.listen((MouseEvent e) {
9       InputElement emailField = querySelector('#email');
10      InputElement passwordField = querySelector('#password');
11      String username = emailField.value;
12      @S("Top") String guess = passwordField.value;
13      if (login.login(username, guess)) {
14        querySelector('#title').text = "Welcome";
15      } else {
16        querySelector('#title').text = "Bad credentials";
17      }
18    });
19  }
20 }
21 }
```

Information flow error in argument. Expected a subtype of [hashCode: Bot]. Found Top.

## Ejemplo: Sistema de autenticación web

```
main.dart x database.dart x
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Home {
6   void render() {
7     Login login = new Login();
8     querySelector('#button').onClick.listen((MouseEvent e) {
9       InputElement emailField = querySelector('#email');
10      InputElement passwordField = querySelector('#password');
11      String username = emailField.value;
12      @S("Hash") String guess = passwordField.value;
13      if (login.login(username, guess)) {
14        querySelector('#title').text = "Welcome";
15      }
16      else {
17        querySelector('#title').text = "Bad credentials";
18      }
19    });
20  }
21 }
```

## Ejemplo: Sistema de autenticación web

```
main.dart x index.html x database.dart x
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <link href="https://fonts.googleapis.com/icon?family=Material+Icon"
6   <link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare
7   <meta charset="utf-8">
8   <meta http-equiv="X-UA-Compatible" content="IE=edge">
9   <meta name="viewport" content="width=device-width, initial-scale=1">
10  <meta name="scaffolded-by" content="https://github.com/google/sta
11  <title>Secure Login</title>
12  <link rel="stylesheet" href="styles.css">
13  <link rel="icon" href="favicon.ico">
14  <script defer src="main.dart.js"></script>
15  <style>
```

# Ejemplo: Sistema de autenticación web

## Secure Login

Enter your credentials

Enter your email

Enter your password

[Forgot Password?](#)

LOGIN

**Tabla 1:** Anotación de facetas públicas en identificadores

Con inferencia	Sin inferencia
7	22



Test unitarios en el repositorio del proyecto.

# Conclusión

---

Conexión entre abstracciones de tipo y relaciones de orden de etiquetas de seguridad.

Formalización de inferencia.

Extensión al subconjunto soportado de Dart.

Características de la extensión para ambientes de desarrollo.

Extensión a polimorfismo.

Extensión a polimorfismo.  
(lista parametrizada)



# Preguntas

Cosas extra por posibles preguntas