DESCLASIFICACIÓN BASADA EN TIPOS EN DART

IMPLEMENTACIÓN Y ELABORACIÓN DE HERRAMIENTAS DE INFERENCIA

Matías Meneses Cortés

Contenidos

Control de flujo de información

Protección de confidencialidad

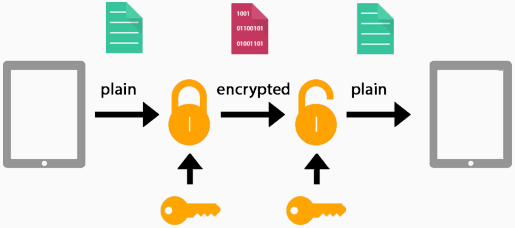
Protección de confidencialidad

Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.

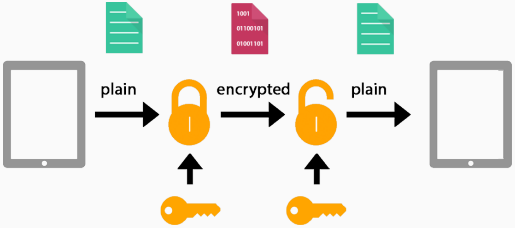
Protección de confidencialidad

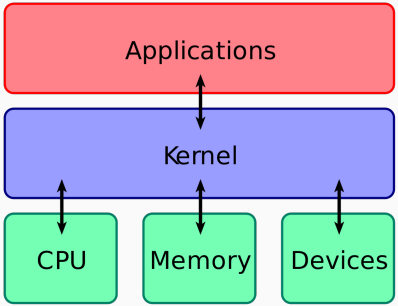
Distintas técnicas de seguridad en distintas capas de comunicación.



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";
}
```

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

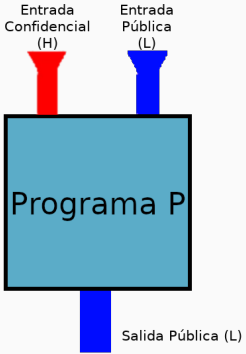

```
String@L book(String@L username, int@L date, int@H cardNumber) {
  return sendToGoogle(username, date, cardNumber);
}

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

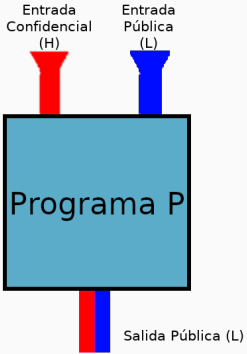

```
String@L book(String@L username, int@L date, int@H cardNumber) {
  return sendToHotel(username, date, cardNumber);
}

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

Problema con no-interferencia

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

¡**No** cumple con no-interferencia!

Desclasificación

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

Problema con desclasificación

```
declassify(password)
```

Problema con desclasificación

```
declassify(password)
```

¡Grave fuga de información!

Desclasificación basada en tipos

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

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`

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";
}
```

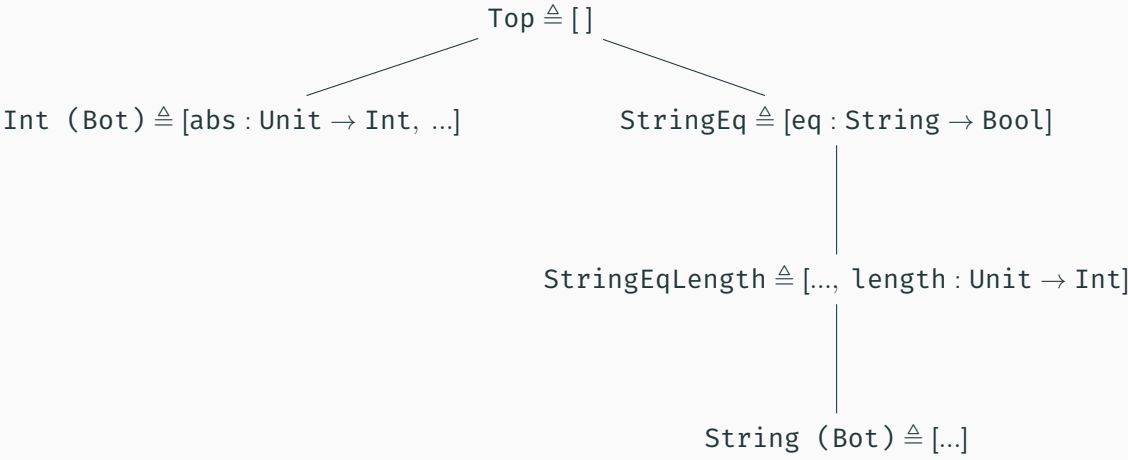
- Tipos de dos facetas `String<StringEq`
- `StringEq = [eq: String<String -> Bool<Bool]`
- `String <: StringEq` (Tipo bien formado)

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

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";
}
```

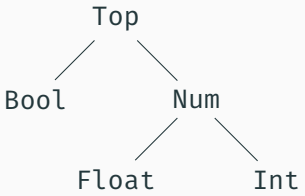
Inferencia de tipos

Inferencia de tipos

```
calculate(c, Int n) {
  if (c) return n*2;
  else return n*0.5;
}
```

Inferencia de tipos

```
calculate(c, Int n) {
  if (c) return n*2;
  else return n*0.5;
}
```



Variables de tipo

```
X calculate(Y c, Int n) {
  if (c) return n*2;
  else return n*0.5;
}
```

Generación de restricciones

```
X calculate(Y c, Int n) {
  if (c) return n*2;
  else return n*0.5;
}
```

Generación de restricciones

```
X calculate(Y c, Int n) {
  if (c) return n*2;
  else return n*0.5;
}
```

1. `Y <: Bool`

Generación de restricciones

```
X calculate(Y c, Int n) {
  if (c) return n*2;
  else return n*0.5;
}
```

1. `Y <: Bool`
2. `Int <: X`

Generación de restricciones

```
X calculate(Y c, Int n) {
  if (c) return n*2;
  else return n*0.5;
}
```

1. `Y <: Bool`
2. `Int <: X`
3. `Float <: X`

Resolución de restricciones

(Mostrar substituciones hasta resolver)

Restricciones sobre subtipos

(codigo anotado con variables de tipo)
(retículo de subtipos)

(restricciones generadas)

Restricciones sobre subtipos

(codigo anotado con variables de tipo)
(retículo de subtipos mostrando meet y join)

(restricciones generadas)

Objetivo de la memoria

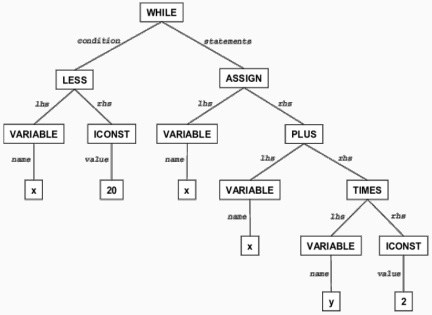
Objetivo de la memoria

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

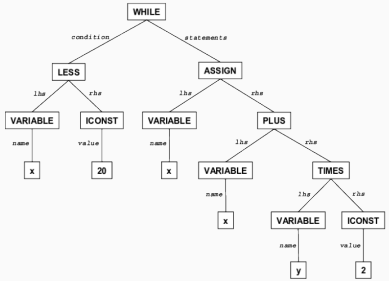
Dart

Dart Analyzer



Abstract Syntax Tree

Analyzer Plugin



Abstract Syntax Tree

Errores y sugerencias de edición

Problema de inferencia a resolver

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

Problema de inferencia a resolver

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código parcialmente tipado 1)

Problema de inferencia a resolver

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código con tipos inferidos 1)

Problema de inferencia a resolver

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código parcialmente tipado 2)

Problema de inferencia a resolver

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código con tipos inferidos 2)

Problema de inferencia a resolver

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código parcialmente tipado 3)

Problema de inferencia a resolver

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código con tipos inferidos 3)

Pasos de la solución

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Pasos de la solución

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Subconjunto soportado de Dart

```
class Foo {
```

Subconjunto soportado de Dart

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

Subconjunto soportado de Dart

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

Subconjunto soportado de Dart

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

Subconjunto soportado de Dart

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

Pasos de la solución

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Declaración de facetas públicas

Uso de anotaciones de Dart para declarar las facetas públicas.

Declaración de facetas públicas

Uso de anotaciones de Dart para declarar las facetas públicas.
(código con `@S("StringEq")`)

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.
(código con `abstract class StringEq`)

Pasos de la solución

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Conversión de tipos de Dart a facetas privadas

(código donde se usan tipos definidos por el usuario y tipos de dart)

Conversión de tipos de Dart a facetas privadas

(código donde se usan tipos definidos por el usuario y tipos de dart, destacando tipo definido por el usuario)

Conversión de tipos de Dart a facetas privadas

(código donde se usan tipos definidos por el usuario y tipos de dart, destacando tipo estático común de dart)

Conversión de tipos de Dart a facetas privadas

(Mostrar operación de convert)

Conversión de tipos de Dart a facetas privadas

(Mostrar operación de convert)
P_{Bi} (algo)

Conversión de tipos de Dart a facetas privadas

(Mostrar operación de convert)
P_{Ai} (algo)

Pasos de la solución

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Gramática de tipos

Gramática de tipos

$$\tau := \alpha \mid \texttt{Obj}(\overline{\texttt{l} : \tau}) \mid \overline{\tau} \to \tau \mid \tau \sqcap \tau \mid \tau \sqcup \tau \mid \texttt{Bot} \mid \texttt{Top}$$

Pasos de la solución

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Generación de restricciones

(Código con invocación a método y variables de tipo) (restricciones generadas)

Generación de restricciones

(Código con expresión de retorno y variables de tipo) (restricciones generadas)

Generación de restricciones

(Código con expresión de asignación y variables de tipo)

(restricciones generadas)

Pasos de la solución

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Resolución de restricciones

Simplificación de restricciones

Resolución de restricciones

Simplificación de restricciones
(Restricciones no simplificadas)

Resolución de restricciones

Simplificación de restricciones
(Restricciones no simplificadas con obvias marcadas)

Resolución de restricciones

Simplificación de restricciones
(Restricciones simplificadas)

Resolución de restricciones

Agrupación de restricciones
(Restricciones no agrupadas)

Resolución de restricciones

Agrupación de restricciones
(Marcar con distinto color las restricciones de un grupo u otro)

Resolución de restricciones

Agrupación de restricciones
(Restricciones agrupadas)

Resolución de restricciones

Unificación: Construcción de tipos
(Restricciones agrupadas)

Resolución de restricciones

Unificación: Construcción de tipos
(Restricciones agrupadas con candidatos a construcción marcados)

Resolución de restricciones

Unificación: Construcción de tipos
(Restricciones con nuevos tipos)

Resolución de restricciones

Unificación: Verificación de restricciones
(Restricciones)

Resolución de restricciones

Unificación: Verificación de restricciones
(Restricciones con relaciones inválidas que provienen de invocación a método marcadas)

Resolución de restricciones

Unificación: Verificación de restricciones
(Restricciones actualizadas luego de comprobación)

Resolución de restricciones

Unificación: Verificación de restricciones
(Restricciones con relaciones inválidas que no provienen de invocación a método marcadas)

Resolución de restricciones

Unificación: Verificación de restricciones
(Error generado por la restricción no válida)

Resolución de restricciones

Unificación: Substitución de restricciones resueltas (Restricciones)

Resolución de restricciones

Unificación: Substitución de restricciones resueltas
(Restricciones con restricciones resueltas marcadas)

Resolución de restricciones

Unificación: Substitución de restricciones resueltas
(Substitución de restricciones)

Unificación: Algoritmo iterativo

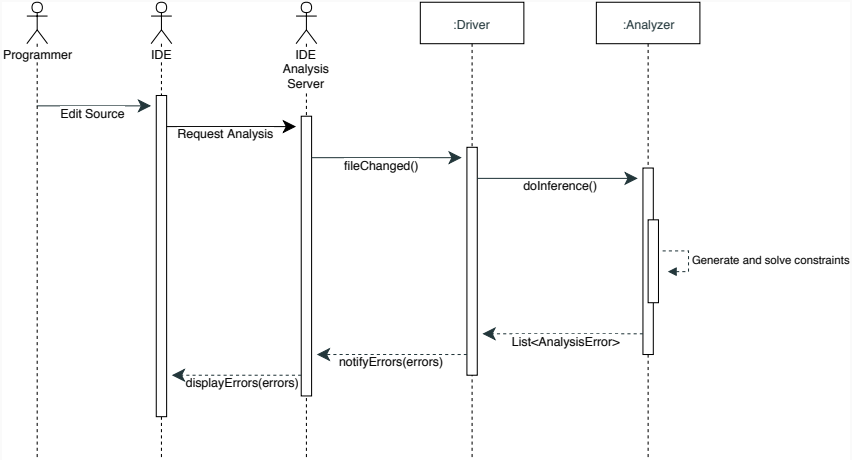
Algoritmo de unificación

(pseudocódigo del algoritmo)

Pasos de la solución

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Extensión para ambientes de desarrollo



Tipos de errores

`SecurityError`

Tipos de errores

`SecurityError`
(código que genera una restricción inválida de esas)

Tipos de errores

`IllFormedTypeError`

Tipos de errores

`IllFormedTypeError`
(código que posee un tipo mal formado)

Tipos de errores

`UndefinedFacetWarning`

Tipos de errores

`UndefinedFacetWarning`
(Código con faceta pública no definida)

Tipos de errores

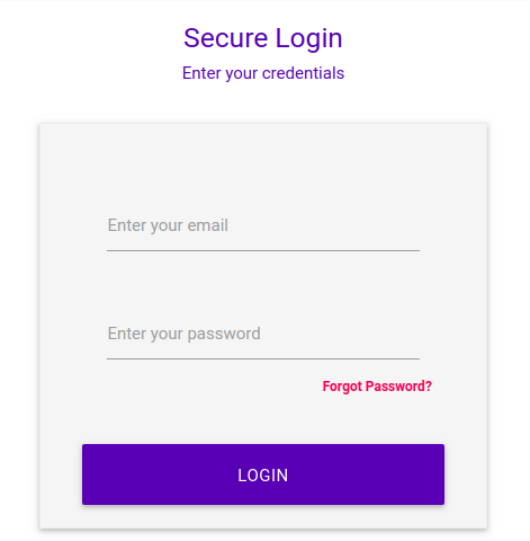
```
InferredFacetInfo
```

Tipos de errores

`InferredFacetInfo`
(código con un tipo inferido)

Validación

Ejemplo: Sistema de autenticación web



Ejemplo: Sistema de autenticación web

(imagen en intelliJ definiendo la base de datos)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ definiendo el metodo login)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ mostrando la inferencia)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ declarando una faceta pública no definida)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ con el warning sobre la faceta no definida)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ usando libreria html)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ agregando Top a variable de contraseña)

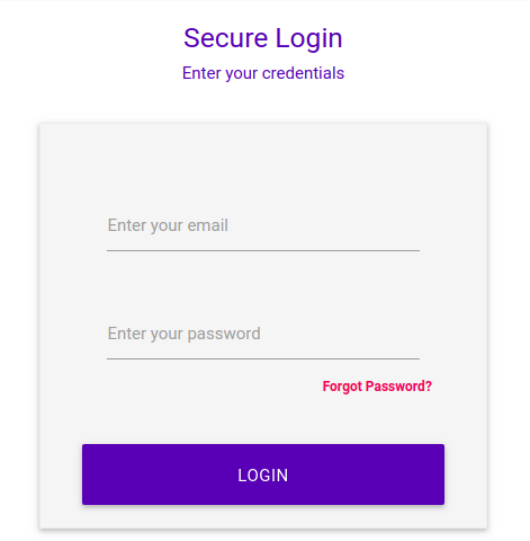
Ejemplo: Sistema de autenticación web

(imagen en intelliJ del error de seguridad)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ corrigiendo el error)

Ejemplo: Sistema de autenticación web



Ejemplo: Sistema de autenticación web

Tabla 1: Anotación de facetas públicas en identificadores

Con inferencia	Sin inferencia
7	22

Comprobación de las reglas del sistema de tipos

Test unitarios en el repositorio del proyecto.

Conclusión

Conclusión

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

Trabajo futuro

Formalización de inferencia.

Trabajo futuro

Extensión al subconjunto soportado de Dart.

Trabajo futuro

Características de la extensión para ambientes de desarrollo.

Trabajo futuro

Extensión a polimorfismo.

Trabajo futuro

Extensión a polimorfismo.
(lista parametrizada)

Preguntas

Apéndice

Cosas extra por posibles preguntas