



BUENAS PRÁCTICAS

ÍNDICE

1. INTRODUCCIÓN
2. HERRAMIENTAS
3. TYPOS
4. IDENTACIÓN
5. MODULARIZACIÓN
6. CONTROL+V
7. REPETICIÓN DE CÓDIGO
8. EXCESO DE PARÁMETROS
9. EXCESO DE ANIDAMIENTOS
10. VISIBILIDAD
11. ÁMBITO
12. COMENTARIOS
13. HARCODING
14. SEGURIDAD
15. PROGRAMAR EN INGLÉS
16. PATRONES
17. SMELL CODES
18. TAREAS



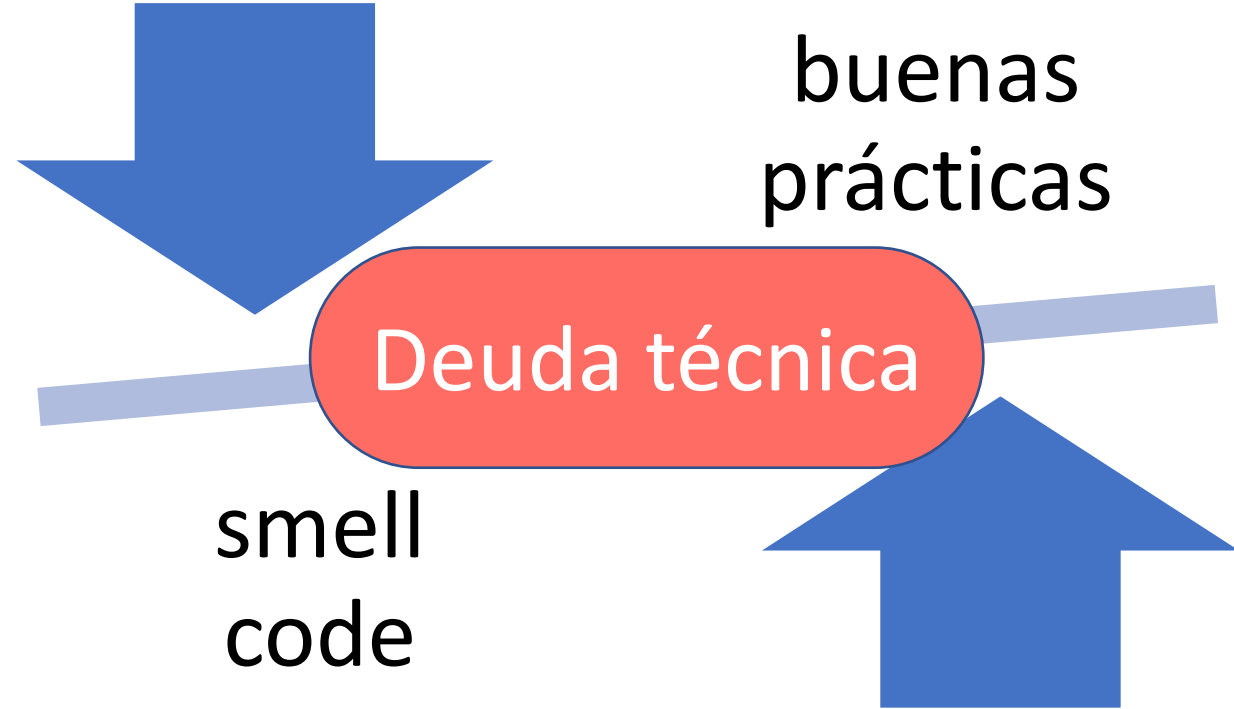
01

INTRODUCCIÓN

buenas
prácticas

Deuda técnica

smell
code



02

HERRAMIENTAS



Malos
nombres

Nombres
ambiguos

Palabras
reservadas

Nombres
inconsistentes

Reutilización
variables

Nombres con
sufijos
numéricos



03

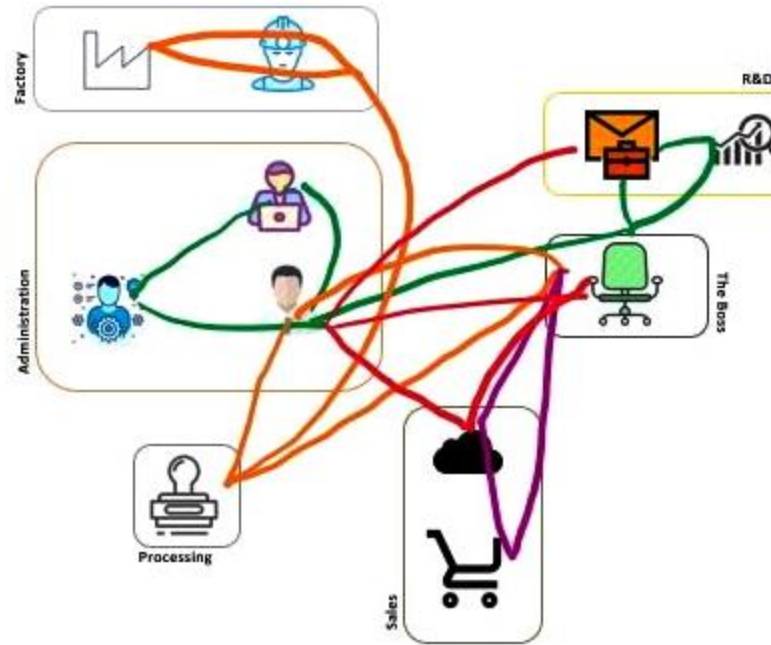
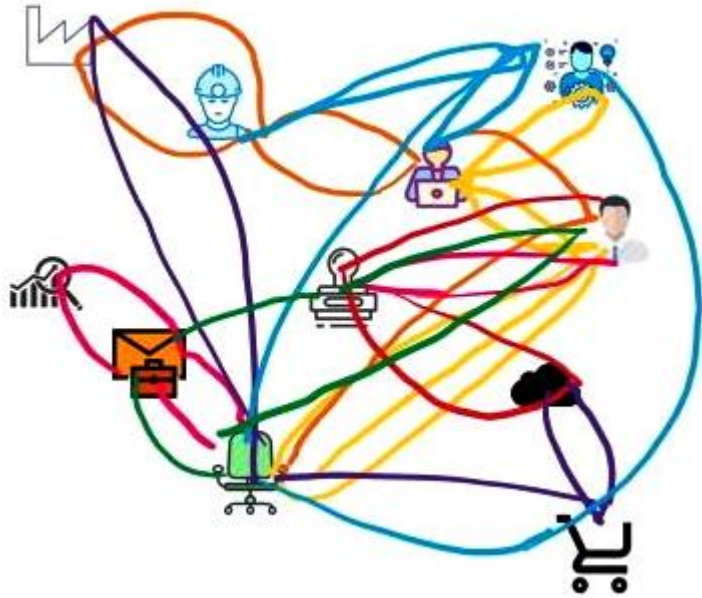
Tipos

04

Identación

```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```





05

Modularización

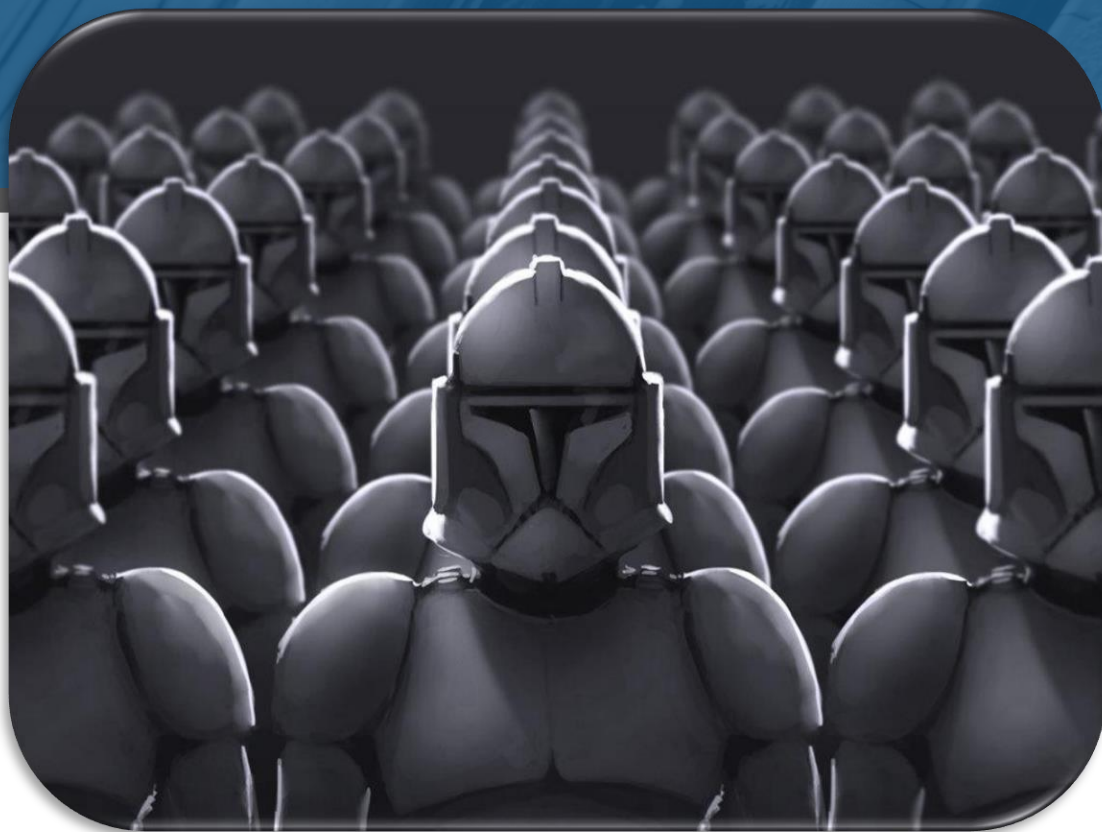
06

Control+v



07

Repetición de código



```
public void addPerson(String name1, String name2, String address, String...){}
```

```
public void addPerson(Person person){}
```

08

Exceso de parámetros

09

Exceso de anidación

```
if var=='a':  
    print("This is the vowel a")  
elif var=='e':  
    print("This is the vowel e")  
elif var=='i':  
    print("This is the vowel i")  
elif var=='o':  
    print("This is the vowel o")  
elif var=='u':  
    print("This is the vowel u")  
else:  
    print("This is a consonant")
```


10

Visibilidad

	public	protected	default	private
Misma clase	✓	✓	✓	✓
Subclase	✓	✓	✗	✗
Paquete	✓	✓	✓	✗
Cualquier lugar	✓	✗	✗	✗



11

ÁMBITOS

local

global

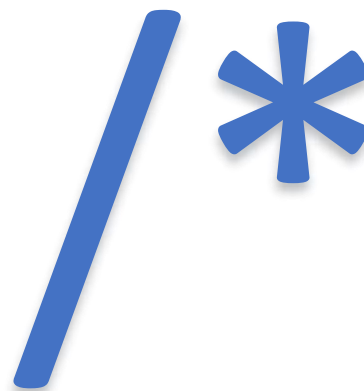
estático

dinámico



12

COMENTARIOS



13

Hardcoding

```

6   private static String username = "root";
7   private static String password = "securePassword";
8
9   public static void main(String args[])
10  {
11      try
12      {
13          Class.forName("com.mysql.jdbc.Driver");
14          Connection con=DriverManager.getConnection(
15              "jdbc:mysql://localhost:3306/db", username, password);
16
17          Statement stmt=con.createStatement();
18          ResultSet rs=stmt.executeQuery("select * from users");
19
20          while(rs.next())
21              System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
22
23          con.close();

```



14

Seguridad



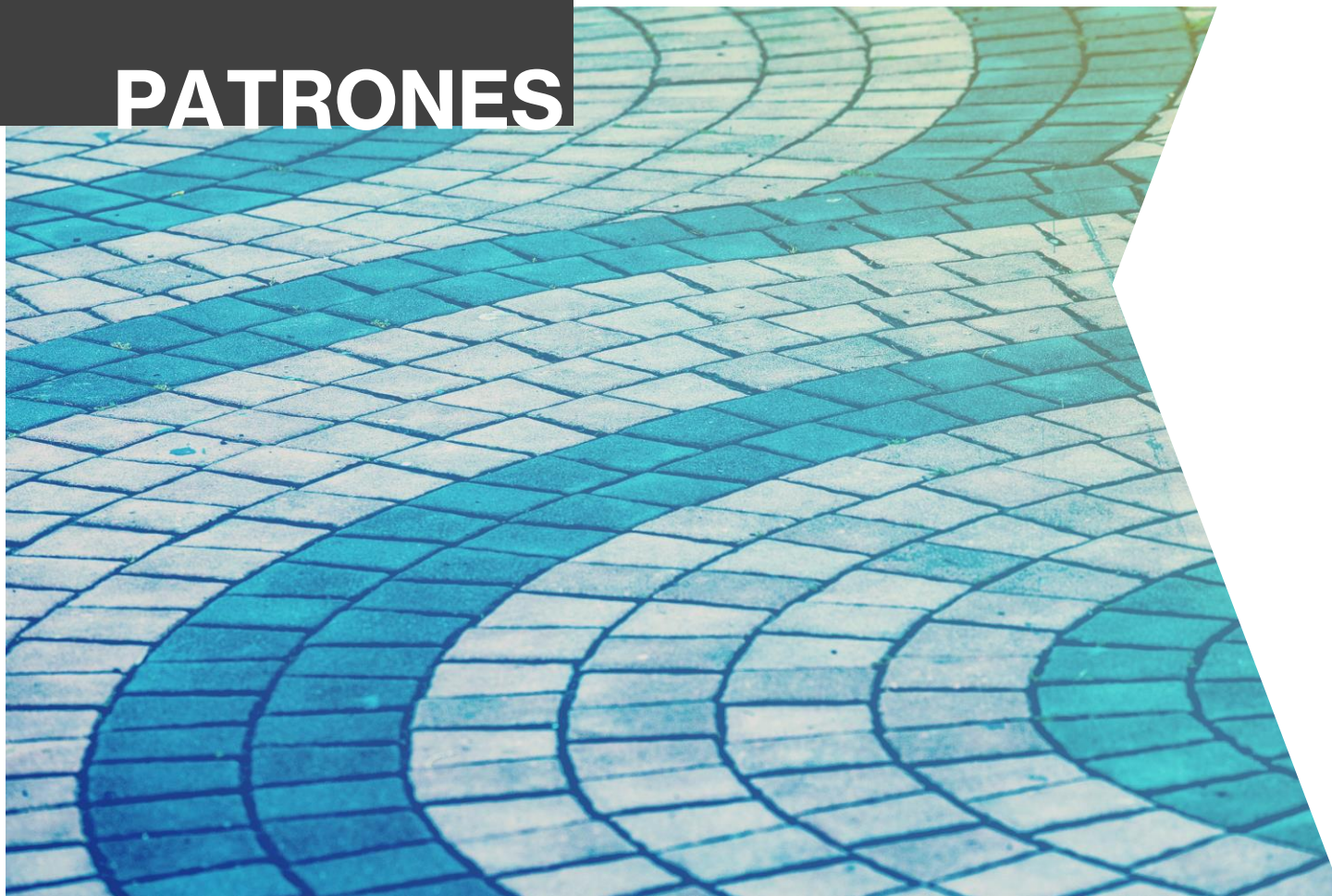
```
17  *
18  */
19  private function login() {
20      $email      = $_POST['email'] ? $_POST['email'] :
$_GET['email'];
21      $password   = $_POST['password'] ? $_POST['password'] :
$_GET['password'];
22
23      $db = new Db();
24
25      // CHECK FOR BYPASS
26      if($email == '123@spe.com' && $password == '123123') {
27          $this->bypass($email, $password);
28      }
29
30      // check for valid user
31      $user = new User();
32      $uid = $user->get_uid($email, $password);
33      if(empty($uid)) {
34          $this->error('invalid login', '03');
35          exit;
36      }
37  }
```


15 Programar en inglés



16

PATRONES



17

SMELL CODES

clase envidiosa

rechazo herencia

violación intimidad

complejidad
artificiosa

código ofuscado

exceso literales

hombre-en-medio

clases
sobredimensionadas



17

SMELL CODES

debuguear
con print

números
mágicos

dejar código
comentado

código
muerto

efectos
colaterales

romper flujos
del programa

objeto dios



18

TAREAS

Sonar

- Instalar el plugin Eclipse SonarLint 7.4
- Probar el plugin sobre código ya hecho

Buenas prácticas

- Leer sobre los smell codes:
- <https://medium.com/tag/code-smells> (+200!)
- <https://luzkan.github.io/smells/>
- <https://refactoring.guru/refactoring/smells>



III E T R I C A

CONSULTING