Código fuente de la aplicación

Código fuente Kernell (Python):

-----data.py-----

```
import pandas as pd
from sklearn.preprocessing import (MinMaxScaler,StandardScaler)


class Data():


    def __init__(self,path=None,data=None,skiprows=None,sep=",",header='infer',index = None):
        if isinstance(path, str):
            self.path = path
            raw_df = pd.read_csv(path,skiprows=skiprows,sep=sep,header="infer")
            self.dataframe = pd.DataFrame(data = raw_df.values, columns = raw_df.columns.values)
            self.columns_names = self.dataframe.columns.tolist()
        elif isinstance(data, pd.DataFrame):
            self.dataframe = pd.DataFrame(data)
        self.shape = self.dataframe.shape
        self.types = self.dataframe.dtypes.tolist()
        self.indexdf = index


    def get_selected_rows(self,index):
        """
        Parameters:
            index: int, list or str


        Returns
            If index type is int return a single row from dataset
```

2

```
        If index type is list return a subset of selected rows from dataset
        If index type is str return a subset of selected rows from dataset
    """
    if isinstance(index, int) and index <= self.shape[0]:
        return self.dataframe.iloc[index]
    elif isinstance(index, list) and max(index) <= self.shape[0]:
        return self.dataframe.iloc[index]
    elif isinstance(index, str):
        raw_index = index.split(":")
        index_slice = list(map(int, raw_index))
        if index_slice[1] <= self.shape[0]:
            return self.dataframe.iloc[index_slice[0]:index_slice[1]]


def get_selected_columns(self,index):
    """
    Parameters:
        index: int, list or str


    Returns
        If index type is int return a single column from dataset
        If index type is list return a subset of selected columns from dataset
        If index type is str return a subset of selected columns from dataset
    """
    if isinstance(index, int) and index <= self.shape[1]:
        return self.dataframe.iloc[:,index]
    elif isinstance(index, list) and max(index) <= self.shape[1]:
        return self.dataframe.iloc[:, index]
    if isinstance(index, str):
```

3

```
            raw_colums = index.split(':')
            column_slice = list(map(int,raw_colums))
            return self.dataframe.iloc[:,column_slice[0]: column_slice[1]]

    def drop_column(self, index_column):
        ""
        if max(index_column) <= self.dataframe.shape[1]:
            self.dataframe = self.dataframe.drop(columns=[self.columns_names[i] for i in index_column])
            self.shape = self.dataframe.shape
            self.columns_names = self.dataframe.columns.tolist()

    def drop_rows(self, index_row):
        if max(index_row) <= self.shape[0]:
            self.dataframe = self.dataframe.drop(index_row)
            self.shape = self.dataframe.shape

    def filter_num(self, index_column, value, filter_type):
        if filter_type == 0:
            return self.dataframe[self.dataframe[self.columns_names[index_column]] > value]
        if filter_type == 1:
            return self.dataframe[self.dataframe[self.columns_names[index_column]] >= value]
        if filter_type == 2:
            return self.dataframe[self.dataframe[self.columns_names[index_column]] < value]
        if filter_type == 3:
            return self.dataframe[self.dataframe[self.columns_names[index_column]] <= value]
        if filter_type == 4:
            return self.dataframe[self.dataframe[self.columns_names[index_column]] == value]
        if filter_type == 5:
```

```
            return self.dataframe[self.dataframe[self.columns_names[index_column]] != value]
        if filter_type == 6:
            return self.dataframe[(self.dataframe[self.columns_names[index_column]] > value[0]) &
(self.dataframe[self.columns_names[index_column]] < value[1])]
        if filter_type == 7:
            return self.dataframe[(self.dataframe[self.columns_names[index_column]] >= value[0]) &
(self.dataframe[self.columns_names[index_column]] <= value[1])]


    def summary(self):
        return self.dataframe.describe()


    def rename_column(self,old,new):
        self.dataframe.rename(columns={old:new})
        self.columns_names = self.dataframe.columns.tolist()


    def scale_column(self,index_column,type):
        desired_column = self.columns_names[index_column]
        data = self.dataframe[desired_column].values.reshape(-1,1)
        if type == 0:
            scaler = MinMaxScaler()
            scaler.fit(data)
            trans_data = scaler.transform(data)
            self.dataframe[desired_column+" Min-Max Scaled"] = trans_data
        elif type == 1:
            scaler = StandardScaler()
            scaler.fit(data)
            trans_data = scaler.transform(data)
            self.dataframe[desired_column+" Standard Scaled"] = trans_data
        self.columns_names = self.dataframe.columns.tolist()
```

5

```
        self.shape = self.dataframe.shape

    def count_na_values(self):
        return self.dataframe.isnull().sum(axis = 0)

    def save_csv(self):
        self.dataframe.to_csv('temp.csv',index=False)

    def info_to_json(self):
        return {
            'columns_names' : self.columns_names,
            'shape' : list(self.shape),
            'index': self.indexdf
        }

    def correlation(self):
        return self.dataframe.corr()
```

-----statistics.py-----

```
import numpy as np
from scipy import stats
from sklearn.linear_model import LinearRegression
from visualization import (qq_normalplot,regression_plot)
import pandas as pd
import matplotlib.pyplot as plt

def shapiro(data):
    shapiro_test = stats.shapiro(data)
    pathqq = qq_normalplot(data)
    print(shapiro_test.statistic)
    print(shapiro_test.pvalue)
    return {
    'statistic': shapiro_test.statistic,
    'p_value': shapiro_test.pvalue,
    'path': pathqq
    }

def linear_regression(X,y):
    x_copy = X.values.reshape(-1, 1)
    model = LinearRegression()
    model.fit(x_copy,y)
    x_min,x_max = x_copy.min(),x_copy.max()
    coef = (model.intercept_, model.coef_[0])
    print(model.intercept_)
    print(model.coef_[0])
    print(model.score(x_copy, y))
    path = regression_plot(x_copy,y,x_max,x_min,coef)
```

7

```
def statistic(data):
    name = data.name
    mean = np.mean(data)
    median = np.median(data)
    var = np.var(data)
    std = np.std(data)
    minv,q1,q2,q3,maxv = data.describe()[3:]
    print(name)
    print(mean)
    print(median)
    print(var)
    print(std)
    print(minv)
    print(q1)
    print(q2)
    print(q3)
    print(maxv)

def svm(x_1,x_2,target,kernel,C):
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
    from sklearn.preprocessing import StandardScaler

    encoder = LabelEncoder()
    encoder.fit(target)
    y = encoder.transform(target)

    X = pd.concat([x_1,x_2], axis = 1)
    X_train, X_test, y_train, y_test = train_test_split(
        X,
```

```
        y,
        test_size= 0.2,
        random_state= 42,
        stratify=y
    )

    sc = StandardScaler()
    sc.fit(X_train.values)
    X_train_std = sc.transform(X_train.values)

    sc = StandardScaler()
    sc.fit(X_test.values)
    X_test_std = sc.transform(X_test)

    from mlxtend.plotting import plot_decision_regions
    from sklearn.svm import SVC

    svm = SVC(kernel = kernel, gamma = 0.2,  C = C).fit(X_train_std,y_train)
    plot_decision_regions(X_train_std, y_train, clf=svm, legend=2)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('SVM')
    plt.savefig('../charts/svm.png')
    print(svm.score(X_test_std, y_test))

def tree(x_1,x_2,target,criterion):
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
    from sklearn.preprocessing import StandardScaler
```

9

```
encoder = LabelEncoder()
encoder.fit(target)
y = encoder.transform(target)

X = pd.concat([x_1,x_2], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size= 0.2,
    random_state= 42,
    stratify=y
)

sc = StandardScaler()
sc.fit(X_train.values)
X_train_std = sc.transform(X_train.values)

sc = StandardScaler()
sc.fit(X_test.values)
X_test_std = sc.transform(X_test)

from mlxtend.plotting import plot_decision_regions
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(criterion=criterion, max_depth= 4, random_state=1)
tree.fit(X_train_std, y_train)

plot_decision_regions(X_train_std, y_train, clf=tree, legend=2)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
```

```
plt.title('Tree')
plt.savefig('../charts/tree.png')
print(tree.score(X_test_std, y_test))

from pydotplus import graph_from_dot_data
from sklearn.tree import export_graphviz
import graphviz

dot_data = export_graphviz(tree,
                           filled=True,
                           rounded=True,
                           class_names=['Setosa',
                                        'Versicolor',
                                        'Virginica'],
                           feature_names=['petal length',
                                          'petal width'],
                           out_file='treeb.png')
graph = graph_from_dot_data(dot_data)
graph.write_png('../charts/btree.png')
```

Código fuente Back-End (MySQL):

```
/*---------------------- Person Table ------------------------*/


CREATE TABLE OPENDATA.PERSON(

PersonID                INT       NOT NULL AUTO_INCREMENT,

PersonName                        CHAR(40) NOT NULL,

PersonLastName                    CHAR(40) NOT NULL,

PersonGender            CHAR(1)  NOT NULL      CHECK (PersonGender IN ('M','F','U')) COMMENT 'Based in ISO/IEC 5218',

        PersonAge                         INT     NOT NULL   CHECK (PersonAge BETWEEN 18 AND 60) COMMENT 'Based
in working age limit',

   PersonCurp                     CHAR(20) NOT NULL   CHECK (PersonCurp REGEXP "[A-Z][A-Z][A-Z][A-Z][0-9][0-9][0-9][0-
9][0-9][0-9][A-Z][A-Z][A-Z][A-Z][A-Z][A-Z][A-Z][0-9]") COMMENT 'Based in XXXX-000000-XXX-XXXX-0 pattern',

PersonRFC                         CHAR(20) NOT NULL   CHECK (PersonRFC REGEXP "[A-Z,Ñ,&]{3,4}[0-9]{2}[0-1][0-9][0-3][0-
9][A-Z,0-9]?[A-Z,0-9]?[0-9,A-Z]?") COMMENT 'Based in XXXX-000000-X-00 pattern',

PersonEmail                       CHAR(40) NOT NULL     CHECK (PersonEmail REGEXP "^[a-zA-Z0-9][a-zA-Z0-9.!#$%&'*+-
/=?^_`{|}~]*?[a-zA-Z0-9._-]?@[a-zA-Z0-9][a-zA-Z0-9._-]*?[a-zA-Z0-9]?\\.[a-zA-Z]{2,63}$") COMMENT 'Based in XXXX..N-@-
domain.com pattern',

PersonPhone                       CHAR(20) NOT NULL   CHECK (PersonPhone REGEXP "[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-
9][0-9][0-9]") COMMENT 'Based in 0000000000 pattern',

   PersonCountry                  CHAR(20) NOT NULL,

PersonCity                        CHAR(20) NOT NULL,

PersonPostalCode        CHAR(10) NOT NULL     CHECK (PersonPostalCode REGEXP "[0-9][0-9][0-9][0-9][0-9]"),

PersonBirthDate         DATE     NOT NULL,

PersonHiringDate        DATE     NOT NULL,


PRIMARY KEY (PersonID),

CONSTRAINT UC_PersonID UNIQUE (PersonID),

CONSTRAINT UC_PersonCURP UNIQUE (PersonCurp),

CONSTRAINT UC_PersonRFC UNIQUE (PersonRFC),

CONSTRAINT UC_PersonEMAIL UNIQUE (PersonEmail),

INDEX PersonName(PersonName),

   INDEX PersonCurp(PersonCurp),
```

```
    INDEX PersonRFC(PersonRFC),
    INDEX PersonEmail(PersonEmail),
    INDEX PersonPhone(PersonPhone),
    INDEX PersonHiringDate(PersonHiringDate),
    CONSTRAINT chk_personaNombre CHECK (PersonName REGEXP '^[a-zA-Z]*$'),
    CONSTRAINT chk_personaApellido CHECK (PersonLastName REGEXP '^[a-zA-Z]*$'),
    CONSTRAINT chk_personaCountry CHECK (PersonCountry REGEXP '^[a-zA-Z]*$'),
    CONSTRAINT chk_personaCity CHECK (PersonCity REGEXP '^[a-zA-Z]*$')

);

/*---------------------- User Table -------------------------*/
CREATE TABLE OPENDATA.USER(
UserID                      INT       NOT NULL AUTO_INCREMENT,
PersonID                    INT               NOT NULL,
UserAccountName                 CHAR(40) NOT NULL,
UserPassword            CHAR(40) NOT NULL,
UserRol                         CHAR(20) NOT NULL     CHECK (UserRol IN ('ADMIN_ROL','ANALYST_ROL','GUEST_ROL'))
COMMENT 'Based in Open Data Rol',
UserPhoto                       LONGBLOB,

PRIMARY KEY (UserID),
CONSTRAINT fk_user_person FOREIGN KEY (PersonID) REFERENCES OPENDATA.PERSON(PersonID)
    ON DELETE CASCADE,
    CONSTRAINT UC_UserID UNIQUE (UserID),
    CONSTRAINT UC_UserAccountName UNIQUE (UserAccountName),
    INDEX UserID(UserID),
    INDEX UserAccountName(UserAccountName),
    INDEX UserRol(UserRol),
    CONSTRAINT chk_userAccountName CHECK (UserAccountName REGEXP '^[a-zA-Z0-9]*$')
```

```
);

/*---------------------- Target Table -------------------------*/

CREATE TABLE OPENDATA.TARGET(

TargetID                        INT        NOT NULL AUTO_INCREMENT,

   PersonID                     INT        NOT NULL,

   UserID                                  INT      NOT NULL,

   TargetPath                   CHAR(50) NOT NULL,


   PRIMARY KEY (TargetID),

CONSTRAINT fk_target_person FOREIGN KEY (PersonID) REFERENCES OPENDATA.PERSON(PersonID) ON DELETE
CASCADE,

CONSTRAINT fk_target_user FOREIGN KEY (UserID) REFERENCES OPENDATA.USER(UserID) ON DELETE CASCADE,

CONSTRAINT UC_TargetID UNIQUE (TargetID),

CONSTRAINT UC_TargetPath UNIQUE (TargetPath),

   INDEX TargetID(TargetID),

   INDEX TargetPath(TargetPath)

   );


/*---------------------- Records Table -------------------------*/

CREATE TABLE RECORDS(

                RecordsID               INT        NOT NULL AUTO_INCREMENT,

   RecordsCategory CHAR(20) NOT NULL,

   RecordsTable    CHAR(10) NOT NULL,

   RecordsIdentity CHAR(60) NOT NULL,

   RecordsAction   CHAR(20) NOT NULL,

   RecordsDetails          CHAR(40) NOT NULL,

   RecordsDate             DATE     NOT NULL,

   PRIMARY KEY(RecordsID)

   );
```

14

```
Select * from records;


/*---------------------------- Insert ----------------------------*/

DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER `INSERTPERSON` BEFORE INSERT ON `PERSON` FOR EACH ROW

  BEGIN

    SET @IDENTITY = NEW.PersonEmail;

  INSERT INTO RECORDS (RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate)
VALUES ('TRIGGER','PERSON',@IDENTITY,'INSERT','A PERSON WAS ADDED',NOW());

END $$


DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER `INSERTUSER` BEFORE INSERT ON `USER` FOR EACH ROW

  BEGIN

    SET @IDENTITY = NEW.UserAccountName;

    IF(NEW.UserRol = 'ADMIN_ROL') THEN

                        INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'INSERT','A USER WAS ADDED AS ADMIN',NOW());

                    ELSE

                        IF(NEW.UserRol = 'ANALYST_ROL') THEN

                                            INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'INSERT','A USER WAS ADDED AS ANALYST',NOW());

                        ELSE

                                IF(NEW.UserRol = 'GUEST_ROL') THEN

                                                                        INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'INSERT','A USER WAS ADDED AS GUEST',NOW());

          END IF;

       END IF;

    END IF;

  END $$
```

```
/*---------------------------- Delete ----------------------------*/

DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER `DELETEPERSON` BEFORE DELETE ON `PERSON` FOR EACH ROW

 BEGIN

    SET @IDENTITY = OLD.PersonEmail;

                INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','PERSON',@IDENTITY,'DELETE','A PERSON WAS DELETED',NOW());

END $$


DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER `DELETEUSER` BEFORE DELETE ON `USER` FOR EACH ROW

 BEGIN

    SET @IDENTITY = OLD.UserAccountName;

    IF(OLD.UserRol = 'ADMIN_ROL') THEN

                    INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'DELETE','A USER WAS DELETED AN WAS ADMIN',NOW());

                ELSE

                    IF(OLD.UserRol = 'ANALYST_ROL') THEN

                                    INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'DELETE','A USER WAS DELETED AS WAS ANALYST',NOW());

                    ELSE

                            IF(OLD.UserRol = 'GUEST_ROL') THEN

                                                    INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'DELETE','A USER WAS DELETED AS WAS GUEST',NOW());

        END IF;

      END IF;

    END IF;

 END $$
```

```
/*-------------------------- UPDATE --------------------------*/


  DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER `UPDATEPERSON` BEFORE UPDATE ON `PERSON` FOR EACH ROW

  BEGIN

     SET @IDENTITY = NEW.PersonEmail;

  INSERT INTO RECORDS (RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate)
VALUES ('TRIGGER','PERSON',@IDENTITY,'UPDATE','A PERSON WAS UPDATED',NOW());

END $$


DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER `UPDATEUSER` BEFORE UPDATE ON `USER` FOR EACH ROW

  BEGIN

     SET @IDENTITY = NEW.UserAccountName;

     IF(NEW.UserRol = 'ADMIN_ROL') THEN

                          INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'UPDATE','A USER WAS UPDATED AN IS ADMIN',NOW());

                    ELSE

                          IF(NEW.UserRol = 'ANALYST_ROL') THEN

                                                INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'UPDATE','A USER WAS UPDATED AS IS ANALYST',NOW());

                          ELSE

                                    IF(NEW.UserRol = 'GUEST_ROL') THEN

                                                                                INSERT INTO RECORDS
(RecordsCategory,RecordsTable,RecordsIdentity,RecordsAction,RecordsDetails,RecordsDate) VALUES
('TRIGGER','USER',@IDENTITY,'UPDATE','A USER WAS UPDATED AS IS GUEST',NOW());

          END IF;

       END IF;

     END IF;

  END $$
```

```
/*---------------------- Logs -------------------------*/

/*CHECAR PERSONAS REPETIDAS*/


DROP TABLE IF EXISTS MOVEMENTS;

DROP TRIGGER IF EXISTS UPDATEMOVEMENTSPERSON;

DROP TRIGGER IF EXISTS UPDATEMOVEMENTSUSER;


CREATE TABLE MOVEMENTS(

		MovementID INT NOT NULL AUTO_INCREMENT,

		MovementInfo CHAR(255) NOT NULL,

		PRIMARY KEY(MovementID)

);


DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER `UPDATEMOVEMENTSPERSON` AFTER UPDATE ON `PERSON` FOR EACH ROW

  BEGIN

   INSERT INTO MOVEMENTS (MovementInfo) VALUES (

		CONCAT(now(),',',user(),',PERSON',',UPDATE',',',OLD.PersonName,',',NEW.PersonName,',',OLD.PersonEmail,',',New.PersonEmail,',',OLD.PersonPhone,',',NEW.PersonPhone)

);

  END $$


DELIMITER $$

CREATE DEFINER=`root`@`localhost` TRIGGER `UPDATEMOVEMENTSUSER` AFTER UPDATE ON `USER` FOR EACH ROW

  BEGIN

   INSERT INTO MOVEMENTS (MovementInfo) VALUES (

		CONCAT(now(),',',user(),',USER',',UPDATE',',',OLD.UserAccountName,',',NEW.UserAccountName,',',OLD.UserPassword,',',New.UserPassword,',',OLD.UserRol,',',NEW.UserRol)

);

  END $$
```

Codigo fuente Back-End (Java):

```
package mx.com.od.csv;

import com.opencsv.CSVReader;
import com.opencsv.CSVWriter;
import java.awt.Point;
import java.io.FileWriter;
import java.io.Reader;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 *
 * @author Antonio Pérez Romero
 */
@AllArgsConstructor
@Data
@NoArgsConstructor
public class Csv implements CsvInterface {

    private String csvName;
    private Point csvDimension;
    private String[] csvColumn;
    private List<String[]> csvRows;
```

```
    @Override
    public List<String[]> csvReadAll(Reader reader) throws Exception {
        List<String[]> list;
        try (CSVReader csvReader = new CSVReader(reader)) {
            list = new ArrayList<>();
            list = csvReader.readAll();
        }
        reader.close();
        return list;
    }

    @Override
    public List<String[]> csvOneByOne(Reader reader) throws Exception {
        List<String[]> list = new ArrayList<>();
        try (CSVReader csvReader = new CSVReader(reader)) {
            String[] line;
            while ((line = csvReader.readNext()) != null) {
                list.add(line);
            }
        }
        reader.close();
        return list;
    }

    @Override
    public void csvWriterOneByOne(List<String[]> stringArray, Path path) throws Exception {
        try (CSVWriter writer = new CSVWriter(new FileWriter(path.toString()))) {
            stringArray.forEach((array) -> {
                writer.writeNext(array);
            });
```

```
    }
}

@Override
public void csvWriterAll(List<String[]> stringArray, Path path) throws Exception {
    try (CSVWriter writer = new CSVWriter(new FileWriter(path.toString()))) {
        writer.writeAll(stringArray);
    }
}

@Override
public String csvFindColumn(int columnIndex) {
    return csvIsColumn(columnIndex) ? this.getCsvColumn()[columnIndex] : "Doesn't exist";
}

@Override
public String[] csvFindRow(int rowIndex) {
    return csvIsRow(rowIndex) ? this.getCsvRows().get(rowIndex) : new String[]{"Doesn't exist"};

}

@Override
public String csvFinCell(Point p) {
    return csvIsCell(p) ? this.getCsvRows().get((int) p.getY())[(int) p.getX()] : "Doesn't exist";
}

@Override
public boolean csvIsColumn(int columnIndex) {
    try {
        return this.getCsvColumn()[columnIndex].length() > 0;
```

```
    } catch (IndexOutOfBoundsException ex) {
        System.out.println(" >> Error (Csv IndexBoundsException |006) :" + ex.getMessage());
    }
    return false;
}

@Override
public boolean csvIsRow(int rowIndex) {
    try {
        return this.getCsvRows().get(rowIndex).length > 0;
    } catch (IndexOutOfBoundsException ex) {
        System.out.println(" >> Error (Csv IndexBoundsException |007) :" + ex.getMessage());
    }
    return false;
}

@Override
public boolean csvIsCell(Point p) {
    return csvIsRow(p.y) && csvIsColumn(p.x);
}

@Override
public String csvChangeCellValue(Point p, String newValue, Path path) {
    if (csvIsCell(p)) {
        for (int i = 0; i < this.getCsvRows().size(); i++) {
            System.out.println(i);
            if (i == p.getY()) {
                String[] row = this.getCsvRows().get(i);
                for (int j = 0; j < row.length; j++) {
                    if (j == p.getX()) {
```

```
                    try {
                        row[j] = newValue;
                        this.csvWriterAll(this.getCsvRows(), path);
                        return "Cell up-to-date";
                    } catch (Exception ex) {
                        System.out.println(" >> Error (Csv IndexBoundsException |008) :" + ex.getMessage());
                    }
                }
            }
        }
    } else {
        return " >> Error (Csv Cell Doesn't exist |009)";
    }
    return "S >> Error (Csv Cell Doesn't exist |009)";
}

@Override
public void csvInfo() {
    System.out.println("Name: " + this.getCsvName());
    System.out.println("Columns: " + Arrays.toString(this.getCsvColumn()));
    System.out.println("Dimension: " + this.getCsvDimension());

    this.getCsvRows().forEach((s) -> {
        System.out.println(Arrays.toString(s));
    });
}

}
```

```
package mx.com.od.encrypt;

import java.io.*;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import java.util.Base64;
import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

public class Encrypt implements Serializable,EncryptInterface {

   private static final long serialVersionUID = 1L;
   private static final String secretKeyAES = "OpenData579$23.1!09d0shcvyu%/$12";
   private static final String saltAES = "OpenData";
   private SecretKey secretKeyTemp = null;

   public Encrypt() {
      try {
         secretKeyTemp = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512")
```

```
                .generateSecret(new PBEKeySpec(secretKeyAES.toCharArray(), saltAES.getBytes(), 65536, 256));
        } catch (NoSuchAlgorithmException | InvalidKeySpecException e) {
            e.printStackTrace(System.out);
        }
    }

    @Override
    public String getSHA256(String data) {
        try {
            byte[] iv = new byte[16];
            IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
            SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
            KeySpec keySpec = new PBEKeySpec(secretKeyAES.toCharArray(), saltAES.getBytes(), 65536, 256);
            secretKeyTemp = secretKeyFactory.generateSecret(keySpec);
            SecretKeySpec secretKey = new SecretKeySpec(secretKeyTemp.getEncoded(), "AES");
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivParameterSpec);
            return Base64.getEncoder().encodeToString(cipher.doFinal(data.getBytes("UTF-8")));
        } catch (UnsupportedEncodingException | InvalidAlgorithmParameterException | InvalidKeyException |
NoSuchAlgorithmException | InvalidKeySpecException | BadPaddingException | IllegalBlockSizeException |
NoSuchPaddingException e) {
            e.printStackTrace(System.out);
            return null;
        }
    }

    @Override
    public String getSHA256Descrypt(String data) {
        byte[] iv = new byte[16];
        try {
```

```
            IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
            SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
            KeySpec keySpec = new PBEKeySpec(secretKeyAES.toCharArray(), saltAES.getBytes(), 65536, 256);
            secretKeyTemp = secretKeyFactory.generateSecret(keySpec);
            SecretKeySpec secretKey = new SecretKeySpec(secretKeyTemp.getEncoded(), "AES");
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(Cipher.DECRYPT_MODE, secretKey, ivParameterSpec);
            return new String(cipher.doFinal(Base64.getDecoder().decode(data)));
        } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException |
InvalidKeySpecException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
            e.printStackTrace(System.out);
            return null;
        }
    }
}
```

Codigo fuente de Front-End (XHTML)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Admin Management</title>
        <meta name="viewport" content="user-scalable=no, width=device-width, initial-scale=1.0, maximum-scale=1.0"/>
        <link href="http://fonts.googleapis.com/css?family=Josefin+Sans" rel="stylesheet" type="text/css"/>
        <link href="http://fonts.googleapis.com/css?family=Poppins" rel="stylesheet" type="text/css"/>
        <meta http-equiv="Expires" content="0"/>
        <meta http-equiv="Last-Modified" content="0"/>
        <meta http-equiv="Cache-Control" content="no-cache, mustrevalidate"/>
```

```
    <meta http-equiv="Pragma" content="no-cache"/>
    <h:outputStylesheet library="css" name="adminStyle.css"/>
  </h:head>
  <h:body>
    <section class="personControl">
      <div class="personHeader">
        <h1 style="font-size: 2rem;">
          <p:commandButton icon="pi pi-arrow-left" action="/OpenData/faces/index.xhtml"
                           styleClass="rounded-button ui-button-secondary" />
          Admin Manager
        </h1>
        <h3>
          Add user register
        </h3>
      </div>
      <!-- Form Add Employee-->
      <h:form class="personControlForm">

        <!-- Form -->
        <div class="personControlForm_inputGroup"><!-- Input Group -->
          <label class="personControlForm_label" for="input-name" >Name:</label>
          <p:inputText id="input-name" class="personControlForm_input" required="true"
                       maxlength="30" size="40"
                       placeholder="Name"
                       value="#{personBean.personaSelected.personName}">
            <p:keyFilter regEx="/[a-zA-z]/i"/>
          </p:inputText>
        </div>

        <div class="personControlForm_inputGroup"><!-- Input Group -->
```

```
            <label class="personControlForm_label" for="input-lastName">Last Name:</label>
            <p:inputText id="input-lastName" class="personControlForm_input" required="true"
                    maxlength="30" size="40"
                    placeholder="Last Name"
                    value="#{personBean.personaSelected.personLastName}">
              <p:keyFilter regEx="/[a-zA-z]/i"/>
            </p:inputText>
          </div>

          <div class="personControlForm_inputGroup">
            <label class="personControlForm_label" for="input-gender">Gender:</label>
            <p:selectOneMenu id="input-gender" value="#{personBean.personaSelected.personGender}"
class="personControlForm_input" required="true"
                    maxlength="30">
              <f:selectItem itemLabel="Women" itemValue="F"/>
              <f:selectItem itemLabel="Men" itemValue="M"/>
              <f:selectItem itemLabel="Other" itemValue="U"/>
            </p:selectOneMenu>
          </div>

          <div class="personControlForm_inputGroup"><!-- Input Group -->
            <label class="personControlForm_label" for="input-age">Age:</label>
            <p:spinner id="input-age" class="personControlForm_input"
value="#{personBean.personaSelected.personAge}" required="true" maxlength="30" size="40"  min="18" max="60"/>
          </div>

          <div class="personControlForm_inputGroup"><!-- Input Group -->
            <label class="personControlForm_label" for="input-curp">Curp:</label>
            <p:inputText id="input-curp" class="personControlForm_input" required="true"
                    maxlength="30" size="40"
```

```
                placeholder="Curp"
                value="#{personBean.personaSelected.personCurp}">
            <p:keyFilter regEx="/[a-zA-z0-9]/i"/>
        </p:inputText>
    </div>
    <div class="personControlForm_inputGroup"><!-- Input Group -->
        <label class="personControlForm_label" for="input-rfc">RFC:</label>
        <p:inputText id="input-rfc" class="personControlForm_input" required="true"
                maxlength="30" size="40"
                placeholder="RFC"
                value="#{personBean.personaSelected.personRFC}">
            <p:keyFilter regEx="/[a-zA-z0-9]/i"/>
        </p:inputText>
    </div>
    <div class="personControlForm_inputGroup"><!-- Input Group -->
        <label class="personControlForm_label" for="input-email">Email:</label>
        <p:inputText id="input-email" class="personControlForm_input" required="true"
                maxlength="30" size="40"
                placeholder="Email"
                value="#{personBean.personaSelected.personEmail}">
            <p:keyFilter regEx="/[a-zA-z0-9@.]/i"/>
        </p:inputText>
    </div>
    <div class="personControlForm_inputGroup"><!-- Input Group -->
        <label class="personControlForm_label" for="input-phone">Phone:</label>
        <p:inputText id="input-phone" class="personControlForm_input" required="true"
                maxlength="30" size="40"
                placeholder="Phone"
                value="#{personBean.personaSelected.personPhone}">
            <p:keyFilter mask="num" />
```

```
    </p:inputText>
</div>
<div class="personControlForm_inputGroup"><!-- Input Group -->
    <label class="personControlForm_label" for="input-country">Country:</label>
    <p:inputText id="input-country" class="personControlForm_input" required="true"
                maxlength="30" size="40"
                placeholder="Country"
                value="#{personBean.personaSelected.personCountry}">
        <p:keyFilter regEx="/[a-zA-z]/i"/>
    </p:inputText>
</div>

<div class="personControlForm_inputGroup"><!-- Input Group -->
    <label class="personControlForm_label" for="input-city">City:</label>
    <p:inputText id="input-city" class="personControlForm_input" required="true"
                maxlength="30" size="40"
                placeholder="City"
                value="#{personBean.personaSelected.personCity}">
        <p:keyFilter regEx="/[a-zA-z]/i"/>
    </p:inputText>
</div>
<div class="personControlForm_inputGroup"><!-- Input Group -->
    <label class="personControlForm_label" for="input-postalCode">Postal Code:</label>
    <p:inputText id="input-postalCode" class="personControlForm_input" required="true"
                maxlength="30" size="40"
                placeholder="Postal Code"
                value="#{personBean.personaSelected.personPostalCode}">
        <p:keyFilter mask="num"/>
    </p:inputText>
</div>
```

```
            <style>
                .ui-inputfield{
                    width: 100%;
                }
            </style>

            <div class="personControlForm_inputGroup"><!-- Input Group -->
                <label class="personControlForm_label" for="input-birthDate">Birth Date:</label>
                <p:datePicker id="input-birthDate" class="personControlForm_input" required="true" pattern="MM/dd/yyyy"
value="#{personBean.personaSelected.personBirthDate}" showButtonBar="true">

                </p:datePicker>
            </div>

            <div class="personControlForm_inputGroup"><!-- Input Group -->
                <label class="personControlForm_label" for="input-hiringDate">Hiring Date:</label>
                <p:datePicker id="input-hiringDate" class="personControlForm_input" required="true" pattern="MM/dd/yyyy"
value="#{personBean.personaSelected.personHiringDate}" showButtonBar="true">

                </p:datePicker>
            </div>

            <div class="personControlForm_inputGroup-submit">
                <p:commandButton class="personControlForm_input-submit" value="Add"
action="#{personBean.addPersonListener()}"/>
            </div>
        </h:form>
    </section>

    <!-- Table List Employees-->
```

```
<section class="personView">
  <h:form id="peopleDataTableForm">
    <p:growl id="msgs" showDetail="true"/>
    <p:dataTable id="personDataTable" class="personViewTable" value="#{personBean.people}" var="person"
          editable="true" rowKey="#{person.personID}" selection="#{personBean.personaSelected}"
          selectionMode="single" scrollable="true" reflow="true" resizeMode="true" resizableColumns="true">

      <p:column class="personViewTable_column" headerText="ID">
        <h:outputText value="#{person.personID}" />
      </p:column>

      <p:column class="personViewTable_column" headerText="Name">
        <p:cellEditor>
          <f:facet name="output">
            <h:outputText value="#{person.personName}"/>
          </f:facet>
          <f:facet name="input">
            <p:inputText value="#{person.personName}" converterMessage="Entrada invalida"/>
          </f:facet>
        </p:cellEditor>
      </p:column>

      <p:column class="personViewTable_column" headerText="Last Name">
        <p:cellEditor>
          <f:facet name="output">
            <h:outputText value="#{person.personLastName}"/>
          </f:facet>
          <f:facet name="input">
            <p:inputText value="#{person.personLastName}"  converterMessage="Entrada invalida"/>
          </f:facet>
```

```
    </p:cellEditor>
</p:column>

<p:column class="personViewTable_column" headerText="Gender" >
    <p:cellEditor>
        <f:facet name="output">
            <h:outputText value="#{person.personGender}"/>
        </f:facet>
        <f:facet name="input">
            <p:inputText value="#{person.personGender}"  converterMessage="Entrada invalida"/>
        </f:facet>
    </p:cellEditor>
</p:column>

<p:column class="personViewTable_column" headerText="Age" >
    <p:cellEditor>
        <f:facet name="output">
            <h:outputText value="#{person.personAge}"/>
        </f:facet>
        <f:facet name="input">
            <p:inputText value="#{person.personAge}"  converterMessage="Entrada invalida"/>
        </f:facet>
    </p:cellEditor>
</p:column>

<p:column class="personViewTable_column" headerText="CURP" >
    <p:cellEditor>
        <f:facet name="output">
            <h:outputText value="#{person.personCurp}"/>
        </f:facet>
```

```
        <f:facet name="input">
          <p:inputText value="#{person.personCurp}"  converterMessage="Entrada invalida"/>
        </f:facet>
      </p:cellEditor>
    </p:column>

    <p:column class="personViewTable_column" headerText="RFC" >
      <p:cellEditor>
        <f:facet name="output">
          <h:outputText value="#{person.personRFC}"/>
        </f:facet>
        <f:facet name="input">
          <p:inputText value="#{person.personRFC}" converterMessage="Entrada invalida"/>
        </f:facet>
      </p:cellEditor>
    </p:column>

    <p:column class="personViewTable_column" headerText="Email" >
      <p:cellEditor>
        <f:facet name="output">
          <h:outputText value="#{person.personEmail}"/>
        </f:facet>
        <f:facet name="input">
          <p:inputText value="#{person.personEmail}" converterMessage="Entrada invalida"/>
        </f:facet>
      </p:cellEditor>
    </p:column>

    <p:column class="personViewTable_column" headerText="Phone" >
      <p:cellEditor>
```

```
        <f:facet name="output">
          <h:outputText value="#{person.personPhone}"/>
        </f:facet>
        <f:facet name="input">
          <p:inputText value="#{person.personPhone}"  converterMessage="Entrada invalida"/>
        </f:facet>
      </p:cellEditor>
    </p:column>

    <p:column class="personViewTable_column" headerText="Country">
      <p:cellEditor>
        <f:facet name="output">
          <h:outputText value="#{person.personCountry}"/>
        </f:facet>
        <f:facet name="input">
          <p:inputText value="#{person.personCountry}" converterMessage="Entrada invalida"/>
        </f:facet>
      </p:cellEditor>
    </p:column>

    <p:column class="personViewTable_column" headerText="City" >
      <p:cellEditor>
        <f:facet name="output">
          <h:outputText value="#{person.personCity}"/>
        </f:facet>
        <!--CHANGE TO FIT-CONTENT-->
        <f:facet name="input">
          <p:inputText value="#{person.personCity}" converterMessage="Entrada invalida"/>
        </f:facet>
      </p:cellEditor>
```

```
    </p:column>

    <p:column class="personViewTable_column" headerText="Postal Code" >
      <p:cellEditor>
        <f:facet name="output">
          <h:outputText value="#{person.personPostalCode}"/>
        </f:facet>
        <f:facet name="input">
          <p:inputText value="#{person.personPostalCode}"  converterMessage="Entrada invalida"/>
        </f:facet>
      </p:cellEditor>
    </p:column>

    <p:column style="width: 20px">
      <p:rowEditor />
    </p:column>

    <p:ajax event="rowEdit" listener="#{personBean.editListener}" update=":peopleDataTableForm:msgs" />

  </p:dataTable>
  <p:contextMenu for="personDataTable">
    <p:menuitem value="Delete" update="peopleDataTableForm:personDataTable"
          icon="pi pi-trash"
          actionListener="#{personBean.deletePersonListener()}" />
    <p:menuitem value="Add User" update="userControlForm" icon="pi pi-user-plus"
          oncomplete="PF('userDialog').show()" />

    <p:menuitem value="See users" update="usersDataTableAdmin" icon="pi pi-search-plus"
          oncomplete="PF('usersDialog').show()"/>
  </p:contextMenu>
```

```
        </h:form>


        <!-- Dialog List Users-->
        <p:dialog id="usersDlg" widgetVar="usersDialog" modal="true" showEffect="fade"
                hideEffect="fade" resizable="false">
            <section class="personView">
                <h:form id="usersDataTableAdmin">
                    <p:growl id="msgs" showDetail="true"/>


                    <p:dataTable id="usersDataTable" class="personViewTable"
value="#{personBean.personaSelected.userList}" var="user" editable="true"  rowKey="#{user.userID}"
                                 selection="#{personBean.userSelect}" selectionMode="single" scrollable="true" reflow="true"
resizeMode="true">


                        <p:column class="personViewTable_column" headerText="Person ID:">
                            <p:cellEditor>
                                <f:facet name="output">
                                    <h:outputText value="#{user.personID.personID}"/>
                                </f:facet>
                                <f:facet name="input">
                                    <p:inputText value="#{user.personID.personID}"/>
                                </f:facet>
                            </p:cellEditor>
                        </p:column>


                        <p:column class="personViewTable_column" headerText="User ID:">
                            <p:cellEditor>
                                <f:facet name="output">
                                    <h:outputText value="#{user.userID}"/>
                                </f:facet>
```

```
                <f:facet name="input">
                    <p:inputText value="#{user.userID}"/>
                </f:facet>
            </p:cellEditor>
        </p:column>

        <p:column class="personViewTable_column" headerText="AccountName:">
            <p:cellEditor>
                <f:facet name="output">
                    <h:outputText value="#{user.userAccountName}"/>
                </f:facet>
                <f:facet name="input">
                    <p:inputText value="#{user.userAccountName}"/>
                </f:facet>
            </p:cellEditor>
        </p:column>

        <p:column class="personViewTable_column" headerText="Rol">
            <p:cellEditor>
                <f:facet name="output">
                    <h:outputText value="#{user.userRol}"/>
                </f:facet>
                <f:facet name="input">
                    <p:inputText value="#{user.userRol}"/>
                </f:facet>
            </p:cellEditor>
        </p:column>
    </p:dataTable>
  </h:form>
</section>
```

```
        </p:dialog>


        <!-- Dialog Add User -->
        <p:dialog id="personaDlg" widgetVar="userDialog" modal="true" showEffect="fade"
                hideEffect="fade" resizable="false">
            <section class="personControl">
                <h:form id="userControlForm" class="userControlForm"><!-- Form -->
                    <p:growl id="msgs" showDetail="true"/>


                    <div class="personControlForm_inputGroup"><!-- Input Group -->
                        <label style="font-size: 2rem;margin-bottom: 2rem;text-align: center; "
class="personControlForm_label">Add User</label>
                    </div>


                    <div class="personControlForm_inputGroup"><!-- Input Group -->
                        <label class="personControlForm_label" for="input-personID">Person ID: </label>
                        <p:inputText id="input-personID" class="personControlForm_input"
                                value="#{personBean.personaSelected.personID}">
                        </p:inputText>
                    </div>


                    <div class="personControlForm_inputGroup"><!-- Input Group -->
                        <label class="personControlForm_label" for="input-username">Username: </label>
                        <p:inputText id="input-username" class="personControlForm_input" required="true"
                                maxlength="30" size="40"
                                value="#{personBean.userSelect.userAccountName}">
                            <p:keyFilter regEx="/[a-zA-z0-9]/i"/>
                        </p:inputText>
                    </div>
```

```
                <div class="personControlForm_inputGroup"><!-- Input Group -->
                  <label class="personControlForm_label" for="input-password">Password: </label>
                  <p:inputText id="input-password" class="personControlForm_input" required="true"
                               maxlength="30" size="40"
                               value="#{personBean.userSelect.userPassword}">
                    <p:keyFilter regEx="/[a-zA-z0-9]/i"/>
                  </p:inputText>
                </div>

                <div class="personControlForm_inputGroup"><!-- Input Group -->
                  <label class="personControlForm_label" for="input-rol">Rol: </label>
                  <p:inputText id="input-rol" class="personControlForm_input" required="true"
                               maxlength="30" size="40"
                               value="#{personBean.userSelect.userRol}">
                    <p:keyFilter regEx="/[a-zA-z]/i"/>
                  </p:inputText>
                </div>
                <div class="personControlForm_inputGroup-submit" style="text-align: center;">
                  <p:commandButton type="submit" class="personControlForm_input-submit" value="Add"
                                   async="true" ajax="true" actionListener="#{personBean.addUserListener}"
                                   update="peopleDataTableForm:personDataTable,usersDataTableAdmin:usersDataTable"
oncomplete="PF('userDialog').hide();" />
                </div>
              </h:form>
            </section>
          </p:dialog>
        </section>
      </h:body>
</html>
```

Codigo fuente Front-End(CSS)

```
:root {
    --lColor: #212121;
    --fontT: Josefin Sans,'Open-sans';  /*Fuente del titulo*/
    --fontS: Poppins, 'Open-sans';      /*Fuente de subtitulo*/
}

html {
    box-sizing: border-box;
}

body {
    display: flex;
    flex-direction: column;
}

.personControl {
    height: 50%;
}

.personHeader{
    margin: 2rem;
    font-family: var(--fontS);
}

.personControlForm {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    column-gap: 2rem;
    grid-row-gap: 1rem;
```

```
    margin: 2rem;
    font-family: var(--fontS);
}

.personControlForm_inputGroup {
    display: grid;
    grid-template-columns: repeat(1, 1fr);
    width: calc(100% - 10px);
    font-family: var(--fontS);
}

.personControlForm_label {
    font-family: var(--fontS);
    font-weight: bold;
}

.personControlForm_input {
    line-height: 1.4rem;
    width: 100%;
}

.personControlForm_inputGroup-submit {
    grid-column: 2/5;
    padding: 1rem 0rem;
    text-align: end;
}

.personControlForm_input-submit {
    width: 49%;
    height: 130%;
```

```
}

body .ui-button {
    background: #607D8B;
    color: #ffffff;
    border: 1px solid #607D8B;
    margin: 0;
    outline: 0 none;
    border-radius: 4px;
    transition: background-color 0.2s, color 0.2s, border-color 0.2s, box-shadow 0.2s;
}

.personControlForm_input-submit:hover {
    cursor: pointer;
}

.personView {
    height: 50%;
    margin: 0 2rem 1rem;
}

/* Responsive */

/* Mobile */

@media only screen and (max-width:690px) {
    .personControlForm {
        display: flex;
        flex-direction: column;
        padding: 1rem;
```

```
        grid-row-gap: .8rem;
        align-items: center;
        font-family: var(--fontS);
    }

    .personControlForm_inputGroup {
        grid-template-columns: repeat(1, 1fr);
        width: 100%;
        font-family: var(--fontS);
        font-weight: bold;
    }

    .personControlForm_inputGroup-submit {
        display: flex;
        width: 50%;
        justify-content: center;
    }

    .personControlForm_input-submit {
        width: 100%;
    }
}

/* Tablet */

@media only screen and (max-width:850px) {
    .personControlForm {
        grid-template-columns: repeat(2, 1fr);
        margin: 1rem;
    }
```

```
    .personControlForm_inputGroup {
        display: flex;
        flex-direction: column;
        width: 90%;
    }

    .personControlForm_inputGroup-submit {
        grid-column: 2/2;
        margin: 0rem .5rem 0rem 0rem;
        padding: 1rem;
    }

    .personControlForm_input-submit {
        width: 60%;
        height: 130%;
    }
    .ui-inputfield{
        width: 100%;
    }
}
```