

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа «Программная инженерия»

СОГЛАСОВАНО
Научный руководитель,
канд. техн. наук, доцент департамента
программной инженерии факультета
компьютерных наук

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной
инженерии, канд. техн. наук

Родригес Р.А. Родригес Залепинос
«25» _____ 09 _____ 2021 г.


_____ В. В. Шилов
« _____ » _____ 2021 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и	
Инв. № подл	

**ПРОГРАММА МОДЕЛИРОВАНИЯ РАСПРОСТРАНЕНИЯ ПОЖАРА С ПОМОЩЬЮ
КЛЕТОЧНЫХ АВТОМАТОВ**

Текст программы

**ЛИСТ УТВЕРЖДЕНИЯ
RU.17701729.05.06-01 12 01-1-ЛУ**

Исполнитель
студент группы БПИ196
 / М. И. Филиппова/
«25» _____ 09 _____ 2021 г.

Москва 2021

УТВЕРЖДЕН
RU.17701729.05.06-01 12 01-1-ЛУ

**ПРОГРАММА МОДЕЛИРОВАНИЯ РАСПРОСТРАНЕНИЯ ПОЖАРА С ПОМОЩЬЮ
КЛЕТОЧНЫХ АВТОМАТОВ**

Текст программы

RU.17701729.05.06-01 12 01-1

Листов 117

<i>Подп. и дата</i>	
<i>Инв. № дубл.</i>	
<i>Взам. инв. №</i>	
<i>Подп. и дата</i>	
<i>Инв. №</i>	

Москва 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1. Подсистема модели	4
1.1 Пакет forest	4
1.1.1 Класс ForestArea	4
1.1.2 Класс ForestCell	24
1.1.3 Класс ForestStates	34
1.3 Пакет urban	39
1.3.1 Класс UrbanArea	39
1.3.2 Класс UrbanCell	56
1.3.3 Класс UrbanStates	64
1.4 Класс GlobalFire	65
1.5 Класс Main	77
2. Подсистема визуализации	79
2.1 Файл UIWindow.py	79
2.2 Файл UIElements	85
2.3 Файл MapView.py	98
2.4 Файл MapEngine.py	104
2.5 Файл Layers.py	109
2.6 Файл Utilities.py	110
2.7 Файл Layers.py	114
2.8 Файл Exceptions.py	115
ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ	117

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

ВВЕДЕНИЕ

Программа состоит из модели и подсистемы визуализации. Далее будет приведен код подсистем.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1. Подсистема модели

1.1 Пакет forest

1.1.1 Класс ForestArea

```

package com.model.forest;

import com.model.urban.UrbanStates;
import com.opencsv.CSVReader;
import com.opencsv.exceptions.CsvValidationException;
import org.apache.commons.lang3.ArrayUtils;
import org.gdal.gdal.*;
import org.gdal.gdalconst.gdalconst;
import org.gdal.ogr.DataSource;
import org.gdal.ogr.ogr;
import org.gdal.ogr.ogrConstants;
import org.gdal.osr.CoordinateTransformation;
import org.gdal.osr.SpatialReference;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.time.LocalDateTime;
import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.Stream;

import com.model.input.InputData;
import com.model.urban.UrbanCell;

public class ForestArea {
    int side;
    int length, width;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

ForestCell[][] cells;
InputData inputData;
String ignitionRasterPath =
"../data/ignition/ignition.tif";

public void setLength(int length) {
    this.length = length;
}

public void setWidth(int width) {
    this.width = width;
}

public void setSpatialReferenceUTM(SpatialReference
spatialReferenceUTM) {
    this.spatialReferenceUTM = spatialReferenceUTM;
}

LocalDateTime currentDate;

ForestStates[][] states;

private SpatialReference spatialReferenceUTM;

public ForestArea(InputData inputData, SpatialReference
spatialReferenceUTM, int length, int width) {
    this.side = inputData.getSide();
    this.inputData = inputData;
    currentDate = inputData.getStart();
    ForestCell.setSide(side);
    this.spatialReferenceUTM = spatialReferenceUTM;
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

this.length                = length;
this.width                  = width;

defineArea(inputData);
defineNeighbours();

gdal.AllRegister();
setElevation(inputData.getElevation());
setSlopes();
setFuel(inputData.getFuel(), inputData.getFuelCodes());
setIgnition(inputData.getIgnition());

```

```

}

```

```

private void setSpreadRates() {
    for (int i = 1; i < width - 1; i++) {
        for (int j = 1; j < length - 1; j++) {
            cells[i][j].initSpreadRates();
        }
    }
}

```

```

private void defineArea(InputData inputData) {
    cells = new ForestCell[width][length];

    var sourceSRS = new SpatialReference();
    sourceSRS.ImportFromEPSG(4326);
    var transform = new CoordinateTransformation(sourceSRS,
spatialReferenceUTM);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

    double[] start =
transform.TransformPoint(inputData.getStartPoint().GetX(),
    inputData.getStartPoint().GetY());

    double x, y;
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            x = start[1] + i * side;
            y = start[0] + j * side;
            cells[i][j] = new ForestCell(x, y);
        }
    }

    states = new ForestStates[width][length];
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            states[i][j] = ForestStates.UNBURNED;
        }
    }
}

private void setIgnition(String ignition) {
    DataSource ign = ogr.Open(ignition);

    var sourceSRS = new SpatialReference();
    sourceSRS.ImportFromEPSG(4326);
    var transform = new CoordinateTransformation(sourceSRS,
spatialReferenceUTM);

    double[] start =
transform.TransformPoint(inputData.getStartPoint().GetX(),

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

        inputData.getStartPoint().GetY());

    for (int i = 0; i < ign.GetLayerCount(); i++) {
        for (int j = 0; j <
            ign.GetLayer(i).GetFeatureCount(); j++) {

            var geom =
                ign.GetLayer(i).GetFeature(j).GetGeometryRef();
            var source = geom.GetSpatialReference();
            var trans = new
                CoordinateTransformation(source, spatialReferenceUTM);
            double[] point;

            if (geom.GetGeometryType() ==
                ogrConstants.wkbPoint) {
                point = trans.TransformPoint(geom.GetX(),
                    geom.GetY());

                int y = (int) Math.round(Math.abs(point[1]
                    - start[1]) / side),

                    x = (int)
                        Math.round(Math.abs(point[0] - start[0]) / side);

                cells[x][y].setState(ForestStates.DEVELOPING);
                states[x][y] = ForestStates.DEVELOPING;
            }

            if (geom.GetGeometryType() ==
                ogrConstants.wkbPolygon) {
                rasterizeIgnition(ignition);
                var dataset =

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

gdal.Open(ignitionRasterPath);

    var band = dataset.GetRasterBand(1);
    int[] presence = new int[1];

    for (int k = 0; i < width; i++) {
        for (int l = 0; j < length; j++) {
            band.ReadRaster(k, length - 1 - l,
1, 1, presence);

            if (presence[0] > 0) {
                states[k][l] =
ForestStates.DEVELOPING;
            }
        }
    }

    band.delete();
    dataset.delete();
}

}

}

private void rasterizeIgnition(String ignition) {
    var ignitionData = ogr.Open(ignition);
    var ignitionLayer = ignitionData.GetLayer(0);

    SpatialReference sourceSrs =
ignitionLayer.GetSpatialRef();
    double[] extent = ignitionLayer.GetExtent();

    double x_res = ((extent[1] - extent[0]) / side);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

double y_res = ((extent[3] - extent[2]) / side);

int xCor = (int) x_res;
int yCor = (int) y_res;

Dataset target_ds = gdal.GetDriverByName("GTiff")
    .Create(ignitionRasterPath, xCor, yCor, 1,
gdalconst.GDT_Byte);

target_ds.SetProjection(sourceSrs.ExportToPrettyWkt());
target_ds.SetGeoTransform(new double[]{extent[0], side,
0, extent[3], 0, -side});

Band band = target_ds.GetRasterBand(1);

int[] intArr = {1};

// Rasterize
gdal.RasterizeLayer(target_ds, intArr, ignitionLayer,
null);

ignitionLayer.delete();
ignitionData.delete();
target_ds.delete();
band.delete();
}

private void defineNeighbours() {
    for (int i = 1; i < width - 1; i++) {
        for (int j = 1; j < length - 1; j++) {
            cells[i][j].setNeighbours(new
ForestCell[]{cells[i - 1][j], cells[i - 1][j + 1],

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
        cells[i][j + 1], cells[i + 1][j + 1],  
        cells[i + 1][j], cells[i + 1][j - 1],  
cells[i][j - 1], cells[i - 1][j - 1]));  
    }  
}  
}
```

```
public void propagate(double minutesLeft, double step,  
LocalDateTime localDateTime) {  
    double newState = 0;  
    currentDate = localDateTime;  
    // поменять погоду  
    if (minutesLeft == 0) {  
        setSpreadRates();  
    }  
  
    for (int i = 2; i < width - 2; i++) {  
        for (int j = 2; j < length - 2; j++) {  
  
            switch (cells[i][j].getState()) {  
                case UNBURNED:  
                    if  
(Arrays.stream(cells[i][j].neighbours)  
                        .anyMatch(x ->  
x.getState().equals(ForestStates.DEVELOPING))) {  
  
                        newState = (cells[i - 1][j -  
1].getSpreadRates()[3] +  
                                cells[i + 1][j -  
1].getSpreadRates()[5] +  
                                cells[i - 1][j +
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
1].getSpreadRates()[1] +  
                                cells[i + 1][j +  
1].getSpreadRates()[7]) * cells[i][j].getFirePeriod() /  
                                Math.sqrt(2) / side +  
cells[i][j].getState().getValue() +  
                                (cells[i][j -  
1].getSpreadRates()[4] +  
                                cells[i -  
1][j].getSpreadRates()[2] +  
                                cells[i +  
1][j].getSpreadRates()[6] +  
                                cells[i][j +  
1].getSpreadRates()[0]) * cells[i][j].getFirePeriod() / side;  
  
                                if (newState >= 1)  
                                    states[i][j] =  
ForestStates.IGNITED;  
                                }  
  
                                break;  
                                case IGNITED:  
                                    if (cells[i][j].getInnerFireTime() ==  
0) {  
                                        var time = side /  
(Math.sqrt(Math.PI) *  
cells[i][j].calculateInternalSpreadRate());  
                                        cells[i][j].setInnerFireTime(time *  
60);  
                                    } else {  
  
cells[i][j].setInnerFireTime(Math.max(0.0,
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

cells[i][j].getInnerFireTime() - step));
        if (cells[i][j].getInnerFireTime()
== 0)

            states[i][j] =

ForestStates.DEVELOPING;
    }
    break;
case DEVELOPING:
    if (cells[i][j].getInnerFireTime() ==
0) {

cells[i][j].setInnerFireTime(cells[i][j].getFirePeriod() * 60);
        } else {

cells[i][j].setInnerFireTime(Math.max(0,
cells[i][j].getInnerFireTime() - step));
            if (cells[i][j].getInnerFireTime()
== 0)

                states[i][j] =

ForestStates.EXTINGUISHING;
        }
        break;
case EXTINGUISHING:
    if (cells[i][j].getInnerFireTime() ==
0) {

cells[i][j].setInnerFireTime(cells[i][j].getFirePeriod() * 60);
        } else {

cells[i][j].setInnerFireTime(Math.max(0,
cells[i][j].getInnerFireTime() - step));

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        if (cells[i][j].getInnerFireTime()
== 0)

            states[i][j] =

ForestStates.BURNED;

        }

        break;

    case BURNED:

        break;

    }

}

}

}

```

```

public void printStatistics() {

    int ignited = 0;

    int burned = 0;

    int developing = 0;

    int unb = 0;

    int ext = 0;

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

        for (int j = 0; j < length; j++) {

            switch (cells[i][j].getState()) {

                case UNBURNED -> {

                    unb++;

                }

                case IGNITED -> {

                    ignited++;

                }

                case DEVELOPING -> {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
        developing++;
    }

    case EXTINGUISHING -> {
        ext++;
    }

    case BURNED -> {
        burned++;
    }
}

}

}

System.out.println("=====" + currentDate.toString()
+ "=====");

System.out.println("IGNITED = " + ignited);
System.out.println("BURNED = " + burned);
System.out.println("DEVELOPING = " + developing);
System.out.println("UNBURNED = " + unb);
System.out.println("EXTINGUISHING = " + ext);
}

public void updateStates() {
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            if (cells[i][j].isIgnitedByUrban()) {
                cells[i][j].setState(ForestStates.IGNITED);
                cells[i][j].makeIgnitedByUrbanDefault();
                states[i][j] = ForestStates.IGNITED;
            }
            cells[i][j].setState(states[i][j]);
        }
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```
    }  
}  
  
    public void setFuel(String path, String fuelCodes) {  
        Map<String, Double> fuelTypesTransition = Stream.of(new  
Object[][]{  
            {"Tree", 0.6},  
            {"Shrub", 0.7},  
            {"Herb", 0.3},  
            {"Agriculture", 1.2},  
            {"Sparse", 0.1} // Barren, Water, Snow-Ice, NA  
-> 0  
        }).collect(Collectors.toMap(data -> (String) data[0],  
data -> (Double) data[1]));  
  
        // read fuel Codes  
        Map<Integer, Double> codes = readFuelCodes(fuelCodes,  
fuelTypesTransition);  
  
        Dataset fuel = gdal.Open(path);  
  
        var paths = generatePaths(path, "fuel.tif");  
  
        fuel = changeProjection(fuel, paths[0]);  
        Dataset modified = changeResolutionAndBorders(fuel,  
paths[1]);  
  
        Band fuelTypes = modified.GetRasterBand(1);  
        int[] value = new int[1];  
  
        for (int i = 0; i < width; i++) {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        for (int j = 0; j < length; j++) {
            fuelTypes.ReadRaster(i, length - 1 - j, 1, 1,
value);

            var val = codes.getDefault(value[0], 0.0) !=
null ?
                codes.getDefault(value[0], 0.0) :
0.0;

            cells[i][j].setFuel(val);
        }
    }

    fuel.delete();
    modified.delete();
}

private Map<Integer, Double> readFuelCodes(String
fuelCodes, Map<String, Double> fuelTypesTransition) {
    Map<Integer, Double> codes = new HashMap<>();
    try {
        var fileReader = new FileReader(fuelCodes);
        var csvReader = new CSVReader(fileReader);
        String[] record;

        // Заголовок.
        record = csvReader.readNext();
        int index = ArrayUtils.indexOf(record, "EVT_LF");

        while ((record = csvReader.readNext()) != null) {
            codes.put(Integer.valueOf(record[0]),
fuelTypesTransition.get(record[index]));

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
    }

    csvReader.close();
    fileReader.close();
}

catch (CsvValidationException | IOException e) {
    e.printStackTrace();
}

return codes;
}

public void setWeatherData(String weatherDataPath) {
    var dataset = gdal.Open(weatherDataPath);
    Band velocity = dataset.GetRasterBand(1);
    Band angle = dataset.GetRasterBand(2);
    Band temperature = dataset.GetRasterBand(3);
    Band humidity = dataset.GetRasterBand(4);

    var temp = new double[1];
    var hum = new double[1];
    var vel = new double[1];
    var ang = new double[1];

    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            temperature.ReadRaster(i, j, 1, 1, temp);
            humidity.ReadRaster(i, j, 1, 1, hum);
            velocity.ReadRaster(i, j, 1, 1, vel);
            angle.ReadRaster(i, j, 1, 1, ang);

            cells[i][j].changeDefaultSpreadRate(temp[0],
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        vel[0], hum[0]);
        cells[i][j].setWindDirection(ang[0]);
        cells[i][j].setWindVelocity(vel[0]);

    }
}

temperature.delete();
humidity.delete();
velocity.delete();
angle.delete();
dataset.delete();
}

private String[] generatePaths(String path, String name) {
    var ind = path.lastIndexOf(File.separator);
    var projectedPath = path.substring(0, ind + 1) +
    "projected_" + name;
    var modifiedPath = path.substring(0, ind + 1) +
    "modified_" + name;
    return new String[]{projectedPath, modifiedPath};
}

public void setElevation(String path) {
    Dataset elevation = gdal.Open(path);

    var paths = generatePaths(path, "elevation.tif");

    elevation = changeProjection(elevation, paths[0]);
    Dataset modified =

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

changeResolutionAndBorders(elevation, paths[1]);

Band heights = modified.GetRasterBand(1);

double[] v = new double[2];
heights.ComputeRasterMinMax(v);
int mean = (int) Math.round((v[0] + v[1]) / 2);

Double[] noDataValue = new Double[1];
heights.GetNoDataValue(noDataValue);

int[] value = new int[1];
for (int i = 0; i < width; i++) {
    for (int j = 0; j < length; j++) {
        heights.ReadRaster(i, length - 1 - j, 1, 1,
value);

        if (value[0] == noDataValue[0]) value[0] = j >
1 ?

(int) cells[i][j - 1].getHeight() :

mean;

        cells[i][j].setHeight(value[0]);

    }
}
elevation.delete();
modified.delete();
}

private void setSlopes() {
    for (int i = 1; i < width - 1; i++) {
        for (int j = 1; j < length - 1; j++) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        cells[i][j].initSlope();
    }
}

}

private Dataset changeResolutionAndBorders(Dataset dataset,
String path) {
    // Изменить размер и разрешение
    var sourceSRS = new SpatialReference();
    sourceSRS.ImportFromEPSG(4326);
    var targetSRS = dataset.GetSpatialRef();

    var ct = new CoordinateTransformation(sourceSRS,
targetSRS);

    var beginning =
ct.TransformPoint(inputData.getStartPoint().GetX(),
inputData.getStartPoint().GetY());

    var finish =
ct.TransformPoint(inputData.getEndPoint().GetX(),
inputData.getEndPoint().GetY());

    Vector<String> options =
        new Vector<>(Arrays.asList("-te",
String.valueOf(beginning[0]), String.valueOf(beginning[1]),
String.valueOf(finish[0]),
String.valueOf(finish[1]),
"-tr", String.valueOf(side),
String.valueOf(side)));

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        var warpOptions = new WarpOptions(options);
        Dataset[] srcData = {dataset};
        Dataset modified = gdal.Warp(path, srcData,
warpOptions);
        return modified;
    }

    private Dataset changeProjection(Dataset dataset, String
path) {
        Vector<String> options = new Vector<>();
        options.add("-t_srs");
        options.add(spatialReferenceUTM.ExportToPrettyWkt());
        WarpOptions warpOptions = new WarpOptions(options);
        Dataset[] srcData = {dataset};
        Dataset projected = gdal.Warp(path, srcData,
warpOptions);
        dataset = gdal.Open(path);

        return dataset;
    }

    public ForestCell[][] getCells() {
        return cells;
    }

    public void propagateInUrban(UrbanCell[][] urbanCells) {
        for (int i = 0; i < width; i++) {
            for (int j = 0; j < length; j++) {
                if
(cells[i][j].getState().equals(ForestStates.DEVELOPING)) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
        cells[i][j].fireSpreadOnUrban(urbanCells,
i, j, width, length);

    }

}

}

}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.1.2 Класс ForestCell

```

package com.model.forest;

import org.gdal.ogr.Geometry;
import org.gdal.ogr.ogr;
import com.model.urban.UrbanCell;
import com.model.urban.UrbanStates;

import java.util.Arrays;

public class ForestCell {
    String geometry;
    private boolean ignitedByUrban = false;
    ForestStates state;
    private double innerFireTime;
    double maxSpreadRate = 0.0;
    double firePeriod = 0.0;
    double fuel;
    double windVelocity;
    double windDirection;
    double height;
    double spreadRateDefault;
    ForestCell[] neighbours; // N NE E SE S SW W NW
    double slope;
    double[] spreadRates;
    static int side;

    public boolean isIgnitedByUrban() {
        return ignitedByUrban;
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
public String getGeometry() {  
    return geometry;  
}  
  
public ForestCell(double x, double y) {  
    state = ForestStates.UNBURNED;  
    Geometry poly = calculateGeometry(x, y);  
    this.geometry = poly.ExportToWkt();  
}  
  
private Geometry calculateGeometry(double x, double y) {  
    var ring = new Geometry(ogr.wkbLinearRing);  
    ring.AddPoint(x, y);  
    ring.AddPoint(x + side, y);  
    ring.AddPoint(x + side, y + side);  
    ring.AddPoint(x, y + side);  
    ring.AddPoint(x, y);  
  
    var poly = new Geometry(ogr.wkbPolygon);  
    poly.AddGeometry(ring);  
    return poly;  
}  
  
public ForestStates getState() {  
    return state;  
}  
  
public void setState(ForestStates state) {  
    this.state = state;  
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
public double getMaxSpreadRate() {  
    return maxSpreadRate;  
}  
  
public double getFirePeriod() {  
    return firePeriod;  
}  
  
public void setFuel(double fuel) {  
    this.fuel = fuel;  
}  
  
public double getFuel() {  
    return fuel;  
}  
  
public void setWindVelocity(double windVelocity) {  
    this.windVelocity = windVelocity;  
}  
  
public void setWindDirection(double windDirection) {  
    this.windDirection = windDirection;  
}  
  
public double getWindVelocity() {  
    return windVelocity;  
}  
  
public void setHeight(double height) {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        this.height = height;
    }

    public double[] getSpreadRates() {
        return spreadRates;
    }

    public void setNeighbours(ForestCell[] neighbours) {
        this.neighbours = neighbours;
    }

    public static int getSide() {
        return side;
    }

    public static void setSide(int side) {
        ForestCell.side = side;
    }

    public double getHeight() {
        return height;
    }

    public double getWindDirection() {
        return windDirection;
    }

    public void initSlope() {
        var x = Math.ceil((neighbours[1].getHeight() -
neighbours[7].getHeight())
                        + 2 * (neighbours[2].getHeight() -

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

neighbours[6].getHeight())
        + (neighbours[3].getHeight() -
neighbours[5].getHeight())) /
        (8 * side); //(neighbours[2].getHeight() -
neighbours[6].getHeight()) / (2 * side);
        var y = Math.ceil((neighbours[7].getHeight() -
neighbours[5].getHeight())
        + 2 * (neighbours[0].getHeight() -
neighbours[4].getHeight())
        + (neighbours[1].getHeight() -
neighbours[3].getHeight())) /
        (8 * side); //(neighbours[0].getHeight() -
neighbours[4].getHeight()) / (2 * side);

        slope = Math.toDegrees(Math.atan(Math.sqrt(x * x + y *
y)));
    }

    public void changeDefaultSpreadRate(double temperature,
                                        double windVelocity,
                                        double humidity) {
        this.spreadRateDefault = 0.03 * temperature + 0.05 *
windVelocity +
                                0.01 * (100 - humidity) - 0.3;
    }

    public void initSpreadRates() {
        if (spreadRates == null)
            spreadRates = new double[8];
        for (int i = 0; i < 8; i++) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        spreadRates[i] = calculateSpreadRate(i);
    }

    maxSpreadRate =
Arrays.stream(spreadRates).max().getAsDouble();

    if (maxSpreadRate > 0) firePeriod = 0.45 * side /
maxSpreadRate;

}

private double calculateSpreadRate(int i) {
    // N NE E SE S SW W NW
    double wind = switch (i) {
        case 0 -> Math.exp(0.1783 * windVelocity *
Math.cos(Math.toRadians(windDirection - 180)));
        case 1 -> Math.exp(0.1783 * windVelocity *
Math.cos(Math.toRadians(windDirection - 135)));
        case 2 -> Math.exp(0.1783 * windVelocity *
Math.cos(Math.toRadians(windDirection - 90)));
        case 3 -> Math.exp(0.1783 * windVelocity *
Math.cos(Math.toRadians(windDirection - 225)));
        case 4 -> Math.exp(0.1783 * windVelocity *
Math.cos(Math.toRadians(windDirection)));
        case 5 -> Math.exp(0.1783 * windVelocity *
Math.cos(Math.toRadians(windDirection + 45)));
        case 6 -> Math.exp(0.1783 * windVelocity *
Math.cos(Math.toRadians(windDirection + 90)));
        case 7 -> Math.exp(0.1783 * windVelocity *
Math.cos(Math.toRadians(windDirection + 225)));
        default -> 1;
    };
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
int sign = 1;
//if (getHeight() > neighbours[i].getHeight()) sign = -
1;

double sl = Math.exp(sign * 3.533 *
Math.pow(Math.tan(Math.toRadians(slope)
*
Math.abs(Math.cos(Math.toRadians(windDirection))), 1.2));

return spreadRateDefault * fuel * wind * sl;

}

public double calculateInternalSpreadRate() {

int sign = 1;
double sl = Math.exp(sign * 3.533 *
Math.pow(Math.tan(Math.toRadians(slope)
*
Math.abs(Math.cos(Math.toRadians(windDirection))), 1.2));

return spreadRateDefault * fuel * sl * Math.exp(0.1783
* windVelocity);
}

public double getInnerFireTime() {
return innerFireTime;
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

public void setInnerFireTime(double innerFireTime) {
    this.innerFireTime = innerFireTime;
}

public void fireSpreadOnUrban(UrbanCell[][] urbanCells, int
i, int j, int width, int length) {
    var k = getMaxSpreadRate() < 13.1 ? 3 : 4.5;
    var a = (3 * getWindVelocity() / 5 + 3) * k + side *
1.0 / 2;
    var b = -2 * getWindVelocity() / 15 + 3 + side * 1.0 /
2;
    var c = -1 * getWindVelocity() / 15 + 3 + side * 1.0 /
2;

    var t = Math.sqrt(b * (a + c) / 2.0);
    var geom = Geometry.CreateFromWkt(geometry).Centroid();
    double x = geom.GetX(), y = geom.GetY();
    var influence = new Geometry(ogr.wkbLinearRing);
    double[] f = rotatedCoords(x - t, y + c, x, y,
windDirection);
    influence.AddPoint(f[0], f[1]);
    f = rotatedCoords(x + t, y + c, x, y, windDirection);
    influence.AddPoint(f[0], f[1]);
    f = rotatedCoords(x + t, y - a, x, y, windDirection);
    influence.AddPoint(f[0], f[1]);
    f = rotatedCoords(x - t, y - a, x, y, windDirection);
    influence.AddPoint(f[0], f[1]);
    f = rotatedCoords(x - t, y + c, x, y, windDirection);
    influence.AddPoint(f[0], f[1]);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

var poly = new Geometry(ogr.wkbPolygon);
poly.AddGeometry(influence);

int mini = (int) Math.max(0, i - a / side);
int minj = (int) Math.max(0, j - a / side);
int maxi = (int) Math.min(width - 1, i + a / side);
int maxj = (int) Math.min(length - 1, j + a / side);

double ign;
for (int l = mini; l <= maxi; l++) {
    for (int m = minj; m <= maxj; m++) {
        if (urbanCells[l][m] != null
            &&
urbanCells[l][m].getState().equals(UrbanStates.UNBURNED)) {
            var urbanGeom =
Geometry.CreateFromWkt(urbanCells[l][m].getGeometry());
            if (urbanGeom.Intersects(poly)) {
                ign = urbanCells[l][m].getMaterial() *
urbanCells[l][m].getWeather()
                    *
urbanGeom.Intersection(poly).Area() / urbanGeom.Area();
                if (k == 3)
                    ign *= 0.4;

urbanCells[l][m].addIgnitionProbability(1 - ign);

            }
        }
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
}

    private double[] rotatedCoords(double pointX, double
pointY,

                                double originX, double
originY, double angle) {
        var x = Math.cos(Math.toRadians(angle)) * (pointX -
originX)
            + Math.sin(Math.toRadians(angle)) * (pointY -
originY) + originX;
        var y = -Math.sin(Math.toRadians(angle)) * (pointX -
originX)
            + Math.cos(Math.toRadians(angle)) * (pointY -
originY) + originY;
        return new double[] {x, y};
    }

    public void becomeIgnited() {
        ignitedByUrban = true;
    }

    public void makeIgnitedByUrbanDefault() {
        ignitedByUrban = false;
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.1.3 Класс ForestStates

```
package com.model.forest;

public enum ForestStates {
    UNBURNED(0),
    IGNITED(1),
    DEVELOPING(2),
    EXTINGUISHING(3),
    BURNED(4);

    ForestStates(int value) {
        this.value = value;
    }

    private final int value;

    public int getValue() {
        return value;
    }
}
```

1.2 Пакет input

1.2.1 Класс InputData

```
package com.model.input;

import org.gdal.ogr.Geometry;
import org.gdal.ogr.ogrConstants;

import java.time.LocalDateTime;
```

```
public class InputData {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
int side;

public String getFuel() {
    return fuel;
}

public String getFuelCodes() {
    return fuelCodes;
}

public String getElevation() {
    return elevation;
}

public int getWeatherPeriod() {
    return weatherPeriod;
}

public String getIgnition() {
    return ignition;
}

String ignition;

String fuel;
String fuelCodes;
String elevation;

public String getBuildingsPath() {
    return buildingsPath;
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
}

String buildingsPath;
int weatherPeriod;

public double getHouseMaterial() {
    return houseMaterial;
}

double houseMaterial;

public InputData(double[] coords, String fuel, String
fuelCodes, String elevation, String meteodata, int
meteoDataChange,
                LocalDateTime start, LocalDateTime finish,
int side, String ignition, String buildingsPath, double
houseMaterial) {
    this.meteodata = meteodata;
    this.start      = start;
    this.finish     = finish;
    this.fuel       = fuel;
    this.fuelCodes  = fuelCodes;
    this.elevation  = elevation;
    this.weatherPeriod = meteoDataChange;
    this.buildingsPath = buildingsPath;
    this.houseMaterial = houseMaterial;

    startPoint = new Geometry(ogrConstants.wkbPoint);
    endPoint = new Geometry(ogrConstants.wkbPoint);
    startPoint.AddPoint(coords[0], coords[1]);
    endPoint.AddPoint(coords[2], coords[3]);
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
        this.ignition = ignition;
        this.side = side;
    }

    public Geometry getStartPoint() {
        return startPoint;
    }

    // coordinates
    private Geometry startPoint;

    public Geometry getEndPoint() {
        return endPoint;
    }

    private Geometry endPoint;

    public String getMeteodata() {
        return meteodata;
    }

    private final String meteodata; // files per time

    public int getSide() {
        return side;
    }

    public LocalDateTime getStart() {
        return start;
    }
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
}

    public LocalDateTime getFinish() {
        return finish;
    }

    // start and end of simulation
    private final LocalDateTime start;
    private final LocalDateTime finish;

}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.3 Пакет urban

1.3.1 Класс UrbanArea

```
package com.model.urban;

import com.model.forest.ForestCell;
import com.model.input.InputData;
import org.gdal.gdal.Band;
import org.gdal.gdal.Dataset;
import org.gdal.gdal.WarpOptions;
import org.gdal.gdal.gdal;
import org.gdal.gdalconst.gdalconst;
import org.gdal.ogr.*;
import org.gdal.osr.CoordinateTransformation;
import org.gdal.osr.SpatialReference;

import java.io.File;
import java.util.*;

public class UrbanArea {
    int width, length;
    int side;
    InputData inputData;
    SpatialReference spatialReferenceUTM;
    String areaVectorPath =
    "..\\data\\buildings\\urban_area.shp";
    String areaRasterPath = "..\\data\\buildings\\buildings.tif";
    UrbanCell[][] urbanCells;
    UrbanStates[][] states;
    Random random = new Random();
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

public UrbanCell[][] getUrbanCells() {
    return urbanCells;
}

public UrbanArea(InputData inputData, SpatialReference
spatialReferenceUTM, int length, int width) {
    this.width                = width;
    this.length               = length;
    this.inputData            = inputData;
    this.spatialReferenceUTM  = spatialReferenceUTM;
    urbanCells               = new UrbanCell[width][length];
    states                   = new
UrbanStates[width][length];
    this.side                 = inputData.getSide();
    UrbanCell.material        = inputData.getHouseMaterial();

    if (inputData.getIgnition().endsWith("osm"))
        extractBuildings(inputData, spatialReferenceUTM);
    rasterizeBuildingMap();
    initUrbanCells();
}

private void initUrbanCells() {
    var dataset = gdal.Open(areaRasterPath);
    var paths = generatePaths(areaRasterPath, "urban.tif");
    dataset = changeProjection(dataset, paths[0]);
    dataset = changeResolutionAndBorders(dataset, paths[1]);

    var sourceSRS = new SpatialReference();
    sourceSRS.ImportFromEPSG(4326);
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

var transform = new CoordinateTransformation(sourceSRS,
spatialReferenceUTM);

double[] start =
transform.TransformPoint(inputData.getStartPoint().GetX(),
inputData.getStartPoint().GetY());

double x, y;

var band = dataset.GetRasterBand(1);
int[] presence = new int[1];
for (int i = 0; i < width; i++) {
    for (int j = 0; j < length; j++) {
        band.ReadRaster(i, length - 1 - j, 1, 1,
presence);

        x = start[1] + i * side;
        y = start[0] + j * side;

        if (presence[0] > 0) {
            urbanCells[i][j] = new UrbanCell(x, y, side);
            states[i][j] = UrbanStates.UNBURNED;
        }
    }
}

band.delete();
dataset.delete();
}

private void rasterizeBuildingMap() {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

var urbanData = ogr.Open(areaVectorPath);
var urbanLayer = urbanData.GetLayer(0);

SpatialReference sourceSrs = urbanLayer.GetSpatialRef();
double[] extent = urbanLayer.GetExtent();

double x_res = ((extent[1] - extent[0]) / side);
double y_res = ((extent[3] - extent[2]) / side);

int xCor = (int) x_res;
int yCor = (int) y_res;

Dataset target_ds = gdal.GetDriverByName("GTiff")
    .Create(areaRasterPath, xCor, yCor, 1,
gdalconst.GDT_Byte);

target_ds.SetProjection(sourceSrs.ExportToPrettyWkt());
target_ds.SetGeoTransform(new double[]{extent[0], side,
0, extent[3], 0, -side});
Band band = target_ds.GetRasterBand(1);

int[] intArr = {1};

// Rasterize
gdal.RasterizeLayer(target_ds, intArr, urbanLayer, null);

urbanLayer.delete();
urbanData.delete();
target_ds.delete();
band.delete();
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

/* public void initIgnition(String path) {

}*/

public void propagate(double step) {
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            if (urbanCells[i][j] == null)
                continue;

            switch (urbanCells[i][j].getState()) {
                case IGNITED -> {
                    urbanCells[i][j].developIgnition(step,
states, random, i, j);
                }
                case SLOWDEVELOPING, FULLDEVELOPMENT -> {
                    double a = 3 *
urbanCells[i][j].getWindVelocity() / 5 + 3 + side / 2.0;
                    double b = -2 *
urbanCells[i][j].getWindVelocity() / 15 + 3 + side / 2.0;
                    double c = -1 *
urbanCells[i][j].getWindVelocity() / 15 + 3 + side / 2.0;
                    var t = Math.sqrt(b * (a + c) / 2.0);
                    var geom =
Geometry.CreateFromWkt(urbanCells[i][j].getGeometry()).Centroid()
;
                    double x = geom.GetX(), y = geom.GetY();
                    var influence = new
Geometry(ogr.wkbLinearRing);
                    double[] f = rotatedCoords(x - t, y + c,

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

x, y, urbanCells[i][j].getWindAngle());
        influence.AddPoint(f[0], f[1]);
        f = rotatedCoords(x + t, y + c, x, y,
urbanCells[i][j].getWindAngle());
        influence.AddPoint(f[0], f[1]);
        f = rotatedCoords(x + t, y - a, x, y,
urbanCells[i][j].getWindAngle());
        influence.AddPoint(f[0], f[1]);
        f = rotatedCoords(x - t, y - a, x, y,
urbanCells[i][j].getWindAngle());
        influence.AddPoint(f[0], f[1]);
        f = rotatedCoords(x - t, y + c, x, y,
urbanCells[i][j].getWindAngle());
        influence.AddPoint(f[0], f[1]);

        var influenceArea = new
Geometry(ogr.wkbPolygon);
        influenceArea.AddGeometry(influence);

        int mini = (int) Math.max(0, i - a /
side);

        int minj = (int) Math.max(0, j - a /
side);

        int maxi = (int) Math.min(width, i + a /
side);

        int maxj = (int) Math.min(length, j + a /
side);

        double ign;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        for (int l = mini; l < maxi; l++) {
            for (int m = minj; m < maxj; m++) {
                if (urbanCells[l][m] != null
                    &&
urbanCells[l][m].getState().equals(UrbanStates.UNBURNED)) {
                    var urbanGeom =
Geometry.CreateFromWkt(urbanCells[l][m].getGeometry());
                    if
(urbanGeom.Intersection(influenceArea) != null) {
                        ign =
urbanCells[l][m].getMaterial() * urbanCells[l][m].getWeather()
                        *
urbanGeom.Intersection(influenceArea).Area() / urbanGeom.Area();

                        if
(urbanCells[i][j].getState().equals(UrbanStates.SLOWDEVELOPING))
                            ign *= 0.4;

urbanCells[l][m].addIgnitionProbability(1 - ign);
                    }
                }
            }
        }

        urbanCells[i][j]
            .fireSpreadOnUrban(step, states,
random, i, j);
    }

    case FLASHOVER ->
urbanCells[i][j].developFlashover(step, states, random, i, j);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        case UNBURNED, EXTINGUISHED -> {
        }
    }
}

}

}

}

private double[] rotatedCoords(double pointX, double pointY,
                                double originX, double
originY, double angle) {
    var x = Math.cos(Math.toRadians(angle)) * (pointX -
originX)
        + Math.sin(Math.toRadians(angle)) * (pointY -
originY) + originX;
    var y = -Math.sin(Math.toRadians(angle)) * (pointX -
originX)
        + Math.cos(Math.toRadians(angle)) * (pointY -
originY) + originY;
    return new double[]{x, y};
}

private void extractBuildings(InputData inputData,
SpatialReference spatialReferenceUTM) {
    gdal.AllRegister();
    var sourceSRS = new SpatialReference();
    sourceSRS.ImportFromEPSG(4326);
    var transform = new CoordinateTransformation(sourceSRS,
spatialReferenceUTM);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

double[] start =
transform.TransformPoint(inputData.getStartPoint().GetX(),
inputData.getStartPoint().GetY());

double[] point;
var data = ogr.Open(inputData.getBuildingsPath());
var layer = data.GetLayerByName("multipolygons");
var source = layer.GetSpatialRef();

var trans = new CoordinateTransformation(source,
spatialReferenceUTM);

var driver = gdal.GetDriverByName("ESRI Shapefile");
var dataset = driver.Create(areaVectorPath, 0, 0,
1, gdalconst.GDT_Unknown, (String[]) null);
var dataLayer = dataset.CreateLayer("houses",
spatialReferenceUTM, ogrConstants.wkbPolygon);

var id = new FieldDefn("id", ogr.OFTInteger);
dataLayer.CreateField(id);

Feature f;
while ((f = layer.GetNextFeature()) != null) {
    for (int i = 0; i < f.GetFieldCount(); i++) {
        if
("house".equals(f.GetFieldAsString("building"))) {
            var feature = new
Feature(dataLayer.GetLayerDefn());

            var geom = f.GetGeometryRef();
            geom.TransformTo(spatialReferenceUTM);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

        feature.SetGeometry(f.GetGeometryRef());
        feature.SetField("id", f.GetFID());

        dataLayer.CreateFeature(feature);
        feature.delete();
    }
}

layer.delete();
data.delete();
dataset.delete();
dataLayer.delete();

}

private String[] generatePaths(String path, String name) {
    var ind = path.lastIndexOf(File.separator);
    var projectedPath = path.substring(0, ind + 1) +
    "projected_" + name;
    var modifiedPath = path.substring(0, ind + 1) +
    "modified_" + name;
    return new String[]{projectedPath, modifiedPath};
}

private Dataset changeResolutionAndBorders(Dataset dataset,
String path) {
    // Изменить размер и разрешение

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

var sourceSRS = new SpatialReference();
sourceSRS.ImportFromEPSG(4326);
var targetSRS = dataset.GetSpatialRef();

var ct = new CoordinateTransformation(sourceSRS,
targetSRS);

var beginning =
ct.TransformPoint(inputData.getStartPoint().GetX(),
inputData.getStartPoint().GetY());
var finish =
ct.TransformPoint(inputData.getEndPoint().GetX(),
inputData.getEndPoint().GetY());

Vector<String> options =
    new Vector<>(Arrays.asList("-te",
String.valueOf(beginning[0]), String.valueOf(beginning[1]),
String.valueOf(finish[0]),
String.valueOf(finish[1]),
"-tr",
String.valueOf(inputData.getSide()),
String.valueOf(inputData.getSide())));

var warpOptions = new WarpOptions(options);
Dataset[] srcData = {dataset};
Dataset modified = gdal.Warp(path, srcData, warpOptions);
return modified;
}

private Dataset changeProjection(Dataset dataset, String
path) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

Vector<String> options = new Vector<>();
options.add("-t_srs");
options.add(spatialReferenceUTM.ExportToPrettyWkt());
WarpOptions warpOptions = new WarpOptions(options);
Dataset[] srcData = {dataset};
Dataset projected = gdal.Warp(path, srcData,
warpOptions);
dataset = gdal.Open(path);

return dataset;
}

public void propagateInForest(ForestCell[][] cells) {
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            if (urbanCells[i][j] == null)
                continue;

            if
(urbanCells[i][j].getState().equals(UrbanStates.SLOWDEVELOPING)
||

urbanCells[i][j].getState().equals(UrbanStates.FULLDEVELOPMENT))
                urbanCells[i][j].fireSpreadOnForest(cells, i,
j, width, length);
        }
    }
}
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

public void updateStates() {
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            if (urbanCells[i][j] != null) {
                urbanCells[i][j].setState(states[i][j]);

                if
(urbanCells[i][j].getState().equals(UrbanStates.UNBURNED)) {
                    if
(urbanCells[i][j].getIgnitionProbability() > 0) {

                        if (random.nextDouble() <= (1 -
urbanCells[i][j].getIgnitionProbability())) {

urbanCells[i][j].setState(UrbanStates.IGNITED);
                            System.out.println("Ignited!");
                            states[i][j] =
UrbanStates.IGNITED;
                        }

urbanCells[i][j].setIgnitionProbability(1.0);
                    }
                }
            }
        }
    }
}

public void printUrbanStatistics() {
    int ignited = 0, unburned = 0, sldevelop = 0, fulldevelop

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

= 0,

```

        exting = 0, flash = 0;
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            if (states[i][j] == null)
                continue;
            switch(states[i][j]){
                case UNBURNED -> {
                    unburned++;
                }
                case IGNITED -> {
                    ignited++;
                }
                case SLOWDEVELOPING -> {
                    sldevelop++;
                }
                case FULLDEVELOPMENT -> {
                    fulldevelop++;
                }
                case FLASHOVER -> {
                    flash++;
                }
                case EXTINGUISHED -> {
                    exting++;
                }
            }
        }
    }
}

```

System.out.println("++++URBAN CELLS++++");

System.out.println("UNBURNED = " + unburned);

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

System.out.println("IGNITED = " + ignited);
System.out.println("SLOW DEVELOP = " + sldevelop);
System.out.println("FULL DEVELOP = " + fulldevelop);
System.out.println("FLASHOVER = " + flash);
System.out.println("EXTINGUISHED = " + exting);
}

public void setWeatherData(String weatherDataPath) {
    var dataset = gdal.Open(weatherDataPath);
    Band velocity = dataset.GetRasterBand(1);
    Band angle = dataset.GetRasterBand(2);
    Band humidity = dataset.GetRasterBand(4);

    double h;
    var hum = new double[1];
    var vel = new double[1];
    var ang = new double[1];

    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            if (urbanCells[i][j] != null) {
                humidity.ReadRaster(i, j, 1, 1, hum);
                velocity.ReadRaster(i, j, 1, 1, vel);
                angle.ReadRaster(i, j, 1, 1, ang);

                urbanCells[i][j].setWindAngle(ang[0]);
                urbanCells[i][j].setWindVelocity(vel[0]);

                if (hum[0] < 30) {
                    h = 1.0;
                } else if (hum[0] < 60) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
        h = 0.8;
    } else h = 0.4;

    if (urbanCells[i][j] != null)
        urbanCells[i][j].setWeather(h);

    }

}

humidity.delete();
velocity.delete();
angle.delete();
dataset.delete();
}

}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.3.2 Класс UrbanCell

```
package com.model.urban;

import com.model.forest.ForestCell;
import com.model.forest.ForestStates;
import org.gdal.ogr.Geometry;
import org.gdal.ogr.ogr;

import java.util.*;

public class UrbanCell {
    double weather;
    String geometry;
    double windVelocity;
    double windAngle;
    UrbanStates state;
    double side;

    public double getWeather() {
        return weather;
    }

    public String getGeometry() {
        return geometry;
    }

    public UrbanCell(double x, double y, double side) {
        this.state = UrbanStates.UNBURNED;
        this.side = side;
        Geometry poly = calculateGeometry(x, y);
        this.geometry = poly.ExportToWkt();
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

    }

    private Geometry calculateGeometry(double x, double y) {
        var ring = new Geometry(ogr.wkbLinearRing);
        ring.AddPoint(x, y);
        ring.AddPoint(x + side, y);
        ring.AddPoint(x + side, y + side);
        ring.AddPoint(x, y + side);
        ring.AddPoint(x, y);

        var poly = new Geometry(ogr.wkbPolygon);
        poly.AddGeometry(ring);

        return poly;
    }

    public double getWindVelocity() {
        return windVelocity;
    }

    public double getSide() {
        return side;
    }

    public double getIgnitionProbability() {
        return ignitionProbability;
    }

    public void setIgnitionProbability(double

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

ignitionProbability) {
    this.ignitionProbability = ignitionProbability;
}

double ignitionProbability = 1.0;

public double getWindAngle() {
    return windAngle;
}

public UrbanStates getState() {
    return state;
}

double[] coords;

public void setState(UrbanStates state) {
    this.state = state;
}

public double getMaterial() {
    return material;
}

public double getInnerTime() {
    return innerTime;
}

public void setInnerTime(double innerTime) {
    this.innerTime = innerTime;
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

    }

    double innerTime = 0;

    public void setMaterial(double material) {
        this.material = material;
    }

    public void setWeather(double weather) {
        this.weather = weather;
    }

    public void setWindVelocity(double windVelocity) {
        this.windVelocity = windVelocity;
    }

    public void setWindAngle(double windAngle) {
        this.windAngle = windAngle;
    }

    public Geometry calculateAreaOfInterest() {
        var a = 3 * windVelocity / 5 + 3 + side / 2;
        var pt = Geometry.CreateFromWkt(geometry).Centroid();
        return pt.Buffer(a);
    }

    public void fireSpreadOnUrban(double step,
                                   UrbanStates[][] states,
    Random rand, int i, int j) {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        if (innerTime == 0) {
            if (state.equals(UrbanStates.SLOWDEVELOPING))
                innerTime = (rand.nextDouble() * 3 + 5) * 60;
            else {
                if (material == 1.0) {
                    innerTime = (rand.nextDouble() * 10 + 10) *
60;

                } else if (material == 0.8)
                    innerTime = (rand.nextDouble() * 10 + 20) *
60;

                else
                    innerTime = (rand.nextDouble() * 10 + 30) *
60;

            }
        } else {
            innerTime -= step;
            if (innerTime <= 0) {
                innerTime = 0;
                states[i][j] =
UrbanStates.values()[state.getValue() + 1];
            }
        }
    }

    public void developIgnition(double step, UrbanStates[][]
states, Random rand, int i, int j) {
        if (innerTime == 0) {
            innerTime = (rand.nextDouble() * 2 + 4) * 60;
        } else {
            innerTime -= step;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        if (innerTime <= 0) {
            innerTime = 0;
            states[i][j] = UrbanStates.SLOWDEVELOPING;
        }
    }
}

public void developFlashover(double step, UrbanStates[][]
states, Random rand, int i, int j) {
    if (innerTime == 0) {
        if (material == 1.0) {
            innerTime = (rand.nextDouble() * 10 + 20) * 60;
        } else if (material == 0.8)
            innerTime = (rand.nextDouble() * 10 + 30) * 60;
        else
            innerTime = (rand.nextDouble() * 10 + 50) * 60;
    } else {
        innerTime -= step;
        if (innerTime <= 0) {
            innerTime = 0;
            states[i][j] = UrbanStates.EXTINGUISHED;
        }
    }
}

static double material = 1.0;

public void fireSpreadOnForest(ForestCell[][] cells, int i,
int j, int width, int length) {
    double a = 3 * getWindVelocity() / 5 + 3 + side / 2;
    double b = -2 * getWindVelocity() / 15 + 3 + side / 2;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

double c = -1 * getWindVelocity() / 15 + 3 + side / 2;

var t = Math.sqrt(b * (a + c) / 2.0);
var geom = Geometry.CreateFromWkt(geometry).Centroid();
double x = geom.GetX(), y = geom.GetY();
var influence = new Geometry(ogr.wkbLinearRing);
double[] f = rotatedCoords(x - t, y + c, x, y,
windAngle);
influence.AddPoint(f[0], f[1]);
f = rotatedCoords(x + t, y + c, x, y, windAngle);
influence.AddPoint(f[0], f[1]);
f = rotatedCoords(x + t, y - a, x, y, windAngle);
influence.AddPoint(f[0], f[1]);
f = rotatedCoords(x - t, y - a, x, y, windAngle);
influence.AddPoint(f[0], f[1]);
f = rotatedCoords(x - t, y + c, x, y, windAngle);
influence.AddPoint(f[0], f[1]);

var influenceArea = new Geometry(ogr.wkbPolygon);
influenceArea.AddGeometry(influence);

int mini = (int) Math.max(0, i - a / side);
int minj = (int) Math.max(0, j - a / side);
int maxi = (int) Math.min(width, i + a / side);
int maxj = (int) Math.min(length, j + a / side);

for (int l = mini; l < maxi; l++) {
    for (int m = minj; m < maxj; m++) {
        if (cells[l][m].getState() !=

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

ForestStates.UNBURNED)

        continue;

        var forestGeom =
Geometry.CreateFromWkt(cells[l][m].getGeometry());
        if (influenceArea.Intersect(forestGeom))
            cells[l][m].becomeIgnited();
    }
}

    private double[] rotatedCoords(double pointX, double
pointY,

                                double originX, double
originY, double angle) {
        var x = Math.cos(Math.toRadians(angle)) * (pointX -
originX)
            + Math.sin(Math.toRadians(angle)) * (pointY -
originY) + originX;
        var y = -Math.sin(Math.toRadians(angle)) * (pointX -
originX)
            + Math.cos(Math.toRadians(angle)) * (pointY -
originY) + originY;
        return new double[]{x, y};
    }

    public void addIgnitionProbability(double v) {
        ignitionProbability *= v;
    }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.3.3 Класс UrbanStates

```

package com.model.urban;

public enum UrbanStates {

    UNBURNED(0),
    IGNITED(1),
    SLOWDEVELOPING(2),
    FULLDEVELOPMENT(3),
    FLASHOVER(4),
    EXTINGUISHED(5);

    UrbanStates(int value) {
        this.value = value;
    }

    private final int value;

    public int getValue() {
        return value;
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.4 Класс GlobalFire

```
package com.model;

import com.opencsv.CSVReader;
import com.opencsv.exceptions.CsvValidationException;
import com.model.forest.ForestArea;
import com.model.input.InputData;
import org.gdal.gdal.Band;
import org.gdal.gdal.Dataset;
import org.gdal.gdal.WarpOptions;
import org.gdal.gdal.gdal;
import org.gdal.gdalconst.gdalconst;
import org.gdal.osr.CoordinateTransformation;
import org.gdal.osr.SpatialReference;
import com.model.urban.UrbanArea;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.time.Duration;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Arrays;
import java.util.Vector;

public class GlobalFire {
    InputData inputData;
    ForestArea forest;
    UrbanArea urban;
    SpatialReference spatialReferenceUTM;
    int side;
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
int length;

public SpatialReference getSpatialReferenceUTM() {
    return spatialReferenceUTM;
}

public int getLength() {
    return length;
}

public int getWidth() {
    return width;
}

public LocalDateTime getCurrentDate() {
    return currentDate;
}

int width;
LocalDateTime currentDate;

public GlobalFire(InputData inputData) {
    this.inputData = inputData;
    this.side      = inputData.getSide();

    initSpatialReference();
    defineAreaSize();

    currentDate = inputData.getStart();
    forest      = new ForestArea(inputData,
    spatialReferenceUTM, length, width);
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        urban          = new UrbanArea(inputData,
spatialReferenceUTM, length, width);

        setWeather(inputData.getMeteodata(), 0);

    }

    private void setWeather(String weather, int number) {
        FileReader file = null;
        try {
            file = new FileReader(weather);

            var csvReader = new CSVReader(file);
            String[] record;
            csvReader.skip(number);
            int ind = weather.lastIndexOf("/");
            if (ind == -1)
                ind = weather.lastIndexOf("\\");

            String dir = weather.substring(0, ind + 1);

            record = csvReader.readNext();

            String weatherPath = mergeWeatherData(dir, record);
            forest.setWeatherData(weatherPath);
            urban.setWeatherData(weatherPath);

            csvReader.close();
        }
        catch (CsvValidationException | IOException e) {
            e.printStackTrace();
        }
    }

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

    }

    private String mergeWeatherData(String dir, String[]
record) {
        String output = dir + "weather" + currentDate

        .format(DateTimeFormatter.ofPattern("yyyy_MM_dd_HH_mm")) +
        ".tif";

        var dataset =
gdal.GetDriverByName("GTiff").Create(output,
        width, length, 4, gdalconst.GDT_Float64);

dataset.SetProjection(spatialReferenceUTM.ExportToPrettyWkt());
        var sourceSRS = new SpatialReference();
        sourceSRS.ImportFromEPSG(4326);
        var transform = new CoordinateTransformation(sourceSRS,
        spatialReferenceUTM);

        double[] start =
transform.TransformPoint(inputData.getStartPoint().GetX(),
        inputData.getStartPoint().GetY());

        dataset.SetGeoTransform(new double[]{start[0], side, 0,
start[1], 0, -side});

        String projectedName = "wind_" + currentDate

        .format(DateTimeFormatter.ofPattern("yyyy_MM_dd_HH_mm")) +
        "_vel.tif";

        addBandToWeatherDataset(dir + record[1], dataset,
projectedName, 1);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        projectedName = "wind_" + currentDate

        .format(DateTimeFormatter.ofPattern("yyyy_MM_dd_HH_mm")) +
        "_ang.tif";

        addBandToWeatherDataset(dir + record[2], dataset,
        projectedName, 2);

        projectedName = "temp_" + currentDate

        .format(DateTimeFormatter.ofPattern("yyyy_MM_dd_HH_mm")) +
        ".tif";

        addBandToWeatherDataset(dir + record[3], dataset,
        projectedName, 3);

        projectedName = "hum_" + currentDate

        .format(DateTimeFormatter.ofPattern("yyyy_MM_dd_HH_mm")) +
        ".tif";

        addBandToWeatherDataset(dir + record[4], dataset,
        projectedName, 4);

        dataset.delete();
        return output;
    }

    private void addBandToWeatherDataset(String name, Dataset
    dataset, String projectedName, int bandNumber) {
        Dataset originalDataset = gdal.Open(name);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

var paths = generatePaths(name, projectedName);

originalDataset = changeProjection(originalDataset,
paths[0]);
originalDataset =
changeResolutionAndBorders(originalDataset, paths[1]);

Band original = originalDataset.GetRasterBand(1);
Band band = dataset.GetRasterBand(bandNumber);

var data = new double[width];
for (int i = length - 1; i >= 0; i--) {
    original.ReadRaster(0, i, data.length, 1, data);
    band.WriteRaster(0, i, data.length, 1, data);
}

original.delete();
originalDataset.delete();
}

private Dataset changeResolutionAndBorders(Dataset dataset,
String path) {
    // Изменить размер и разрешение
    var sourceSRS = new SpatialReference();
    sourceSRS.ImportFromEPSG(4326);
    var targetSRS = dataset.GetSpatialRef();

    var ct = new CoordinateTransformation(sourceSRS,
targetSRS);

    var beginning =

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

ct.TransformPoint(inputData.getStartPoint().GetX(),
                  inputData.getStartPoint().GetY());

    var finish =
ct.TransformPoint(inputData.getEndPoint().GetX(),
                  inputData.getEndPoint().GetY());

    Vector<String> options =
        new Vector<>(Arrays.asList("-te",
String.valueOf(beginning[0]),
                        String.valueOf(beginning[1]),
                        String.valueOf(finish[0]),
String.valueOf(finish[1]),
                        "-tr", String.valueOf(side),
String.valueOf(side)));

    var warpOptions = new WarpOptions(options);
    Dataset[] srcData = {dataset};
    Dataset modified = gdal.Warp(path, srcData,
warpOptions);

    return modified;
}

private Dataset changeProjection(Dataset dataset, String
path) {
    Vector<String> options = new Vector<>();
    options.add("-t_srs");
    options.add(spatialReferenceUTM.ExportToPrettyWkt());
    WarpOptions warpOptions = new WarpOptions(options);
    Dataset[] srcData = {dataset};
    Dataset projected = gdal.Warp(path, srcData,
warpOptions);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

dataset = gdal.Open(path);

return dataset;
}

private String[] generatePaths(String path, String name) {
    var ind = path.lastIndexOf(File.separator);
    var projectedPath = path.substring(0, ind + 1) +
"projected_" + name;
    var modifiedPath = path.substring(0, ind + 1) +
"modified_" + name;
    return new String[]{projectedPath, modifiedPath};
}

public void propagate() {
    int step = 90;
    double minutesLeft = 0;
    System.out.println(currentDate);

    forest.printStatistics();
    urban.printUrbanStatistics();

    while (currentDate.compareTo(inputData.getFinish()) <
0) {

        forest.propagate(minutesLeft, step, currentDate);
        forest.propagateInUrban(urban.getUrbanCells());
        urban.propagate(step);
        urban.propagateInForest(forest.getCells());

        forest.updateStates();
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        urban.updateStates();

        currentDate = currentDate.plusSeconds(step);
        minutesLeft += (step / 60);
        if (minutesLeft == inputData.getWeatherPeriod()) {
            setWeather(inputData.getMeteodata(),
                       (int)
Duration.between(inputData.getStart(), currentDate).toHours());

            forest.printStatistics();
            urban.printUrbanStatistics();
            presentResult();
            minutesLeft = 0;
        }
    }
    presentResult();

}

private void presentResult() {
    presentForestResults();
}

private void presentForestResults() {
    String path = "..\\data\\result\\result_" + currentDate

.format(DateTimeFormatter.ofPattern("yyyy_MM_dd_HH_mm")) +
".tif";

    Dataset resultData =
gdal.GetDriverByName("GTiff").Create(path,

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        width, length,
        1, gdalconst.GDT_Byte);

    var sourceSRS = new SpatialReference();
    sourceSRS.ImportFromEPSG(4326);
    var ct = new CoordinateTransformation(sourceSRS,
    spatialReferenceUTM);
    var beginning =
    ct.TransformPoint(inputData.getStartPoint().GetX(),
        inputData.getStartPoint().GetY());

    double[] geotransform = {beginning[0], 30, 0.0,
    beginning[1], 0, -30};
    resultData.SetGeoTransform(geotransform);

    resultData.SetProjection(spatialReferenceUTM.ExportToPrettyWkt(
    ));

    Band result = resultData.GetRasterBand(1);

    byte value = 0;
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < length; j++) {
            result.WriteRaster(i, j, 1, 1,
                new byte[] { (byte)
    forest.getCells()[i][j].getState().getValue() });
            if (urban.getUrbanCells()[i][j] != null)
                value = (byte)
    urban.getUrbanCells()[i][j].getState().getValue();
            if (value > 0)

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        result.WriteRaster(i, j, 1, 1, new
byte[] {value});

    }

}

result.delete();
resultData.delete();

}

private void initSpatialReference() {
    this.spatialReferenceUTM = new SpatialReference();
    int zone = (int) Math.round(30 +
inputData.getStartPoint().GetY() / 6);
    spatialReferenceUTM.SetProjCS(String.format("UTM %d
(WGS84)", zone));
    spatialReferenceUTM.SetWellKnownGeogCS("WGS84");
    spatialReferenceUTM.SetUTM(zone);

}

private void defineAreaSize() {
    var sourceSRS = new SpatialReference();
    sourceSRS.ImportFromEPSG(4326);

    var ct = new CoordinateTransformation(sourceSRS,
spatialReferenceUTM);
    var beginning =
ct.TransformPoint(inputData.getStartPoint().GetX(),
inputData.getStartPoint().GetY());
    var finish =

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
ct.TransformPoint(inputData.getEndPoint().GetX(),
inputData.getEndPoint().GetY());
```

```

        width = (int) Math.round(Math.abs(beginning[0] -
finish[0]) / side);
        length = (int) Math.round(Math.abs(beginning[1] -
finish[1]) / side);
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.5 Класс Main

```

package com.model;

import com.model.input.InputData;
import org.gdal.gdal.gdal;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class Main {
    public static void main(String[] args){
        gdal.AllRegister();

        int side = Integer.parseInt(args[0]); //30; // m
        String elevation = args[1];
        //"..\data\elevation\US_DEM2016\US_DEM2016.tif";
        String fuel = args[2];
        //"..\data\US_200EVT\US_200EVT.tif";
        String csvfueltypes =
args[3]; //"..\data\US_200EVT\LF16_EVT_200.csv";
        String weather = args[4];
        //"C:\\Users\\admin\\Documents\\firemodel\\project\\data\\weather\\weather.csv";

        String ignition = args[5];
        //"C:\\Users\\admin\\Documents\\firemodel\\project\\data\\ignition\\ignition.shp";

        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy.MM.dd HH:mm");
        LocalDateTime start =

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

LocalDateTime.parse(args[6].concat(" " + args[7]), formatter);
//LocalDateTime.of(2019, 10,28, 1, 34);

    var finish = LocalDateTime.parse(args[8].concat(" " +
args[9]), formatter);; //LocalDateTime.of(2019, 10, 28, 11, 34);
// 11

    int weatherPeriod = Integer.parseInt(args[10]); //60;
// minutes

    double houseMaterial = Double.parseDouble(args[11]);
//1.0;

    double[] coords = {Double.parseDouble(args[12]),
Double.parseDouble(args[13]),
        Double.parseDouble(args[14]),
Double.parseDouble(args[15])}; //{34.11, -118.50, 34.07, -
118.47};

    var buildings = args[16];
// "C:\\Users\\admin\\Documents\\firemodel\\project\\data\\build
ings\\map.osm";

    var input = new InputData(coords, fuel, csvfueltypes,
elevation, weather, weatherPeriod,
        start, finish, side, ignition, buildings,
houseMaterial);

    var globalFire = new GlobalFire(input);
    globalFire.propagate();
}
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2. Подсистема визуализации

2.1 Файл UIWindow.py

```
from UI.UIElements import UIElement, AddLayerWindow,
ModelSettingsWindow
```

```
WINDOW_SIZE = 0
```

```
class UIWindow:
```

```
    MAIN_WINDOW_OBJECTS = [(QMenuBar, "menubar"), (QHBoxLayout,
"mainContent"), (QListWidget, "listOfLayers")]
```

```
    def __init__(self):
        self.main_window = UIElement(UIWindows.MAIN_WINDOW_OBJECTS,
"UI/MainWindow.ui", QMainWindow())
        self.add_layer_window = AddLayerWindow("UI/AddLayerWindow.ui",
self.main_window, self)
        self.launch_model_window =
ModelSettingsWindow("UI/ModelSettingsWindow.ui", self.main_window,
self)

        self.loading_view = None
        self.web = QWebEngineView()
        self.main_window.elements["mainContent"].addWidget(self.web,
stretch=1)
        self.map_view = MapView(self.web, ui=self)

        self.add_layer_window.initialize()
        self.launch_model_window.initialize()
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

self.initialize_menubar()

self.main_window.elements["listOfLayers"].setContextMenuPolicy(Qt.Custo
mContextMenu)

self.main_window.elements["listOfLayers"].customContextMenuRequested.co
nnect(self.show_layers_context_menu)
    self.main_window.element.show()

def initialize_menubar(self):
    self.main_window.element.findChild(QAction,
"actionNew_project").triggered.connect(self.new_project)
    self.main_window.element.findChild(QAction, "actionExit") \
        .triggered.connect(lambda:
self.main_window.element.close())

    action = self.main_window.element.findChild(QAction,
"actionAdd_raster_layer")
    action.triggered.connect(self.show_add_raster_layer_window)

    action = self.main_window.element.findChild(QAction,
"actionAdd_vector_layer")
    action.triggered.connect(self.show_add_vector_layer_window)

    self.main_window.element.findChild(QAction,
"actionShow_list_of_layers").toggled.connect(
        lambda checked: self.show_layers() if checked else
self.hide_layers())

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        self.main_window.element.findChild(QAction,
"actionStart_model").triggered.connect(self.show_launch_model_window)

    def show_layers_context_menu(self, point):
        if self.main_window.elements["listOfLayers"].itemAt(point):
            layer_name =
self.main_window.elements["listOfLayers"].itemAt(point).text()
            context_menu = QMenu()

            info_action = QAction("Видимость", context_menu)
            info_action.setCheckable(True)
            info_action.setChecked(self.map_view.has_layer(layer_name,
True).is_visible)
            info_action.toggled.connect(lambda checked:
self.map_view.set_visible(layer_name, True) if checked else
self.map_view.set_visible(layer_name, False))

            remove_action = QAction("Удалить слой", context_menu)
            remove_action.triggered.connect(self.remove_layer)

            context_menu.addAction(info_action)
            context_menu.addAction(remove_action)

            point.setY(point.y() + 10)

context_menu.exec(self.main_window.elements["listOfLayers"].mapToGlob
al(point))

    def show_add_raster_layer_window(self):
        self.add_layer_window.show(0)

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

def show_add_vector_layer_window(self):
    self.add_layer_window.show(1)

def show_launch_model_window(self):
    self.launch_model_window.show()

def update_layers_list(self):
    self.main_window.element.findChild(QListWidget,
    "listOfLayers").clear()
    for layer in self.map_view.layers:
        self.main_window.element.findChild(QListWidget,
    "listOfLayers").addItem(layer.name)

def show_layers_list(self):
    width = self.main_window.element.findChild(QFrame,
    "left_side_content").width()

    if width == 30:
        newWidth = 300
        self.main_window.element.findChild(QLabel,
    "label").setText("Слои")
    else:
        newWidth = 30
        self.main_window.element.findChild(QLabel,
    "label").setText("")

    self.main_window.element.animation =
    QPropertyAnimation(self.main_window

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

.element.findChild(QFrame, "left_side_content"),

b"minimumWidth")

        self.main_window.element.animation.setStartValue(width)
        self.main_window.element.animation.setEndValue(newWidth)

        self.main_window.element.animation_group =
QParallelAnimationGroup()

self.main_window.element.animation_group.addAnimation(self.main_windo
w.element.animation)

        self.main_window.element.animation_group.start()

def show_message(self, string, caption, icon, parent=None):
    if parent is None:
        parent = self.main_window.element
    message_box = QMessageBox(parent)
    message_box.setIcon(icon)
    message_box.setText(caption)
    message_box.setInformativeText(string)
    message_box.setWindowTitle(caption)
    message_box.exec_()

def show_layers(self):
    self.main_window.element.findChild(QAction,
"actionShow_list_of_layers").setChecked(True)
    self.show_layers_list()

def hide_layers(self):

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        self.main_window.element.findChild(QAction,
"actionShow_list_of_layers").setChecked(False)
        self.show_layers_list()

    def remove_layer(self):
        if self.main_window.element.findChild(QListWidget,
"listOfLayers").currentItem() is not None:
            self.map_view.remove_layer(
                self.main_window.element.findChild(QListWidget,
"listOfLayers").currentItem().text())
            self.update_layers_list()

    def new_project(self):
        if self.main_window.element.findChild(QFrame,
"left_side_content").width() > 30:
            self.hide_layers()

        self.add_layer_window.hide()

    self.main_window.elements["mainContent"].removeWidget(self.web)
    self.web.deleteLater()

    self.web = QWebEngineView()
    self.main_window.elements["mainContent"].addWidget(self.web,
stretch=1)
    self.map_view = MapView(self.web, ui=self)
    self.update_layers_list()

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.2 Файл UIElements

```

class UIElement:
    def __init__(self, elements, ui_path, element_type):
        self.element = element_type
        uic.loadUi(ui_path, self.element)
        self.elements = dict()
        for element in elements:
            self.elements[element[1]] =
self.element.findChild(element[0], element[1])

class AddLayerWindow(UIElement):
    OBJECTS = [(QPushButton, "add_raster_btn"), (QPushButton,
"choose_raster_file_btn"),
               (QDoubleSpinBox, "east_border"), (QDoubleSpinBox,
"north_border"),
               (QDoubleSpinBox, "west_border"), (QDoubleSpinBox,
"south_border"),
               (QLineEdit, "raster_layer_name"), (QLineEdit,
"raster_path"),
               (QPushButton, "add_vector_btn"), (QPushButton,
"choose_vector_file_btn"),
               (QLineEdit, "vector_layer_name"), (QLineEdit,
"vector_path"), (QTabWidget, "layers_tab")]

    def __init__(self, ui_path, parent, ui):
        self.parent = parent
        self.ui = ui
        super().__init__(AddLayerWindow.OBJECTS, ui_path,
QDialog(self.parent.element))

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

def initialize(self):

self.elements["choose_vector_file_btn"].clicked.connect(self.open_vector_file)

self.elements["add_vector_btn"].clicked.connect(self.add_vector_layer)

self.elements["choose_raster_file_btn"].clicked.connect(self.open_raster_file)

self.elements["add_raster_btn"].clicked.connect(self.add_raster_layer)

def open_vector_file(self):
    options = QFileDialog.Options()
    file_name, _ =
QFileDialog.getOpenFileName(self.parent.element, "Открыть файл", "",
                             "GeoJSON,
Shapefile (*.geojson *.shp)", options=options)
    if file_name:
        self.elements["vector_path"].setText(file_name)

def open_raster_file(self):
    options = QFileDialog.Options()
    file_name, _ =
QFileDialog.getOpenFileName(self.parent.element, "Открыть файл", "",
                             "IMG (*.jpeg *.jpg
*.tif *.bmp *.png)", options=options)
    if file_name:

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

self.elements["raster_path"].setText(file_name)

def show(self, tab=0):
    self.elements["layers_tab"].setCurrentIndex(tab)
    self.elements["vector_layer_name"].setText("")
    self.elements["vector_path"].setText("")
    self.elements["raster_layer_name"].setText("")
    self.elements["raster_path"].setText("")
    self.elements["north_border"].setValue(0.0)
    self.elements["south_border"].setValue(0.0)
    self.elements["east_border"].setValue(0.0)
    self.elements["west_border"].setValue(0.0)
    self.element.show()

def hide(self):
    self.element.hide()

def add_raster_layer(self):
    try:

self.ui.map_view.add_raster_layer(self.elements['raster_layer_name'].
text(),

self.elements['raster_path'].text(),

(self.elements["south_border"].value(),

self.elements["west_border"].value()),

(self.elements["north_border"].value(),

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

self.elements["east_border"].value()))
    except FileOpeningException as ex:
        self.ui.show_message(ex.message, "Ошибка!",
QMessageBox.Critical, self.element)

    except LayerAddingException as ex:
        self.ui.show_message(ex.message, "Ошибка!",
QMessageBox.Critical, self.element)

    else:
        self.hide()
        self.ui.update_layers_list()

def add_vector_layer(self):
    try:

self.ui.map_view.add_vector_layer(self.elements['vector_layer_name'].
text(),

self.elements['vector_path'].text())
    except FileOpeningException as ex:
        self.ui.show_message(ex.message, "Ошибка!",
QMessageBox.Critical, self.element)
    except LayerAddingException as ex:
        self.ui.show_message(ex.message, "Ошибка!",
QMessageBox.Critical, self.element)
    else:
        self.hide()
        self.ui.update_layers_list()

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

class ModelSettingsWindow(UIElement):
    OBJECTS = [(QPushButton, "choose_buildings_btn"),
               (QDoubleSpinBox, "east_border"), (QDoubleSpinBox,
"north_border"),
               (QDoubleSpinBox, "west_border"), (QDoubleSpinBox,
"south_border"),
               (QLineEdit, "buildings_path"), (QDateTimeEdit, "end"),
(QDateTimeEdit, "start"),
               (QPushButton, "choose_fuel_codes_btn"), (QPushButton,
"choose_fuel_data_btn"),
               (QPushButton, "choose_ignition_btn"), (QPushButton,
"choose_relief_data_btn"),
               (QPushButton, "choose_weather_btn"), (QLineEdit,
"fuel_codes_path"),
               (QLineEdit, "fuel_path"), (QComboBox,
"house_material"), (QLineEdit, "ignition_path"),
               (QLineEdit, "relief_path"), (QSpinBox, "side"),
(QPushButton, "start_model"),
               (QSpinBox, "weatherStep"), (QLineEdit,
"weather_path"), (QLineEdit, "layer_name")]

    def __init__(self, ui_path, parent, ui):
        self.parent = parent
        self.ui = ui
        self.material = 1.0
        super().__init__(ModelSettingsWindow.OBJECTS, ui_path,
QDialog(self.parent.element))

    def initialize(self):

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
self.elements["start_model"].clicked.connect(self.launch_model)

self.elements["choose_buildings_btn"].clicked.connect(self.open_osm_file)

        self.elements["choose_fuel_codes_btn"].clicked.connect(lambda
_: self.open_text_file("fuel_codes_path"))
        self.elements["choose_fuel_data_btn"].clicked.connect(lambda
_: self.open_raster_file("fuel_path"))

self.elements["choose_ignition_btn"].clicked.connect(self.open_vector_file)

self.elements["choose_relief_data_btn"].clicked.connect(lambda _:
self.open_raster_file("relief_path"))
        self.elements["choose_weather_btn"].clicked.connect(lambda _:
self.open_text_file("weather_path"))

def show(self):
    self.elements["buildings_path"].setText("")
    self.elements["fuel_codes_path"].setText("")
    self.elements["fuel_path"].setText("")
    self.elements["ignition_path"].setText("")
    self.elements["relief_path"].setText("")
    self.elements["weather_path"].setText("")
    self.elements["layer_name"].setText("")

    self.elements["north_border"].setValue(0.0)
    self.elements["south_border"].setValue(0.0)
    self.elements["east_border"].setValue(0.0)
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

self.elements["west_border"].setValue(0.0)

self.elements["side"].setValue(30)
self.elements["weatherStep"].setValue(60)
self.element.show()

def open_text_file(self, file_path):
    options = QFileDialog.Options()
    file_name, _ =
QFileDialog.getOpenFileName(self.parent.element, "Открыть файл", "",
                             "CSV (*.csv)",
options=options)
    if file_name:
        self.elements[file_path].setText(file_name)

def open_raster_file(self, file_path):
    options = QFileDialog.Options()
    file_name, _ =
QFileDialog.getOpenFileName(self.parent.element, "Открыть файл", "",
                             "GeoTiff (*.tif
*.asc)", options=options)
    if file_name:
        self.elements[file_path].setText(file_name)

def open_vector_file(self):
    options = QFileDialog.Options()
    file_name, _ =
QFileDialog.getOpenFileName(self.parent.element, "Открыть файл", "",
                             "Shapefile
(*.shp)", options=options)

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

if file_name:
    self.elements["ignition_path"].setText(file_name)

def open_osm_file(self):
    options = QFileDialog.Options()
    file_name, _ =
QFileDialog.getOpenFileName(self.parent.element, "Открыть файл", "",
                             "OSM, Shapefile
(*.osm *.shp)", options=options)
    if file_name:
        self.elements["buildings_path"].setText(file_name)

    @staticmethod
    def jarWrapper(*args):
        process = check_output(['java', '-
Djava.library.path=..\model\lib-gdal', '-jar'] + list(args),
stderr=PIPE)

    def launch_model(self):
        try:
            if self.elements["house_material"].currentText() ==
"Смешанный":
                self.material = 0.8
            elif self.elements["house_material"].currentText() ==
"Огнеупорный":
                self.material = 0.6
            else:
                self.material = 1.0

        if not self.check_model_params():

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        args = ['../model/model-1.0.jar',
str(self.elements["side"].value()),
            self.elements["relief_path"].text(),
self.elements["fuel_path"].text(),
            self.elements["fuel_codes_path"].text(),
self.elements["weather_path"].text(),
            self.elements["ignition_path"].text(),

str(self.elements["start"].dateTime().toString("yyyy.MM.dd
HH:mm")).split()[0],

str(self.elements["start"].dateTime().toString("yyyy.MM.dd
HH:mm")).split()[1],

str(self.elements["end"].dateTime().toString("yyyy.MM.dd
HH:mm")).split()[0],

str(self.elements["end"].dateTime().toString("yyyy.MM.dd
HH:mm")).split()[1],

            str(self.elements["weatherStep"].value()),
            str(self.material),
str(self.elements["north_border"].value()),
            str(self.elements["west_border"].value()),
            str(self.elements["south_border"].value()),
str(self.elements["east_border"].value()),
            self.elements["buildings_path"].text()]

        self.jarWrapper(*args)

        self.ui.show_message("Моделирование прошло успешно",

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

"Успешно", QMessageBox.Information)
    except CalledProcessError:
        self.ui.show_message("Ошибка при моделировании",
"Ошибка", QMessageBox.Critical)
    except LayerAddingException as ex:
        self.ui.show_message(ex.message, "Ошибка",
QMessageBox.Critical)
    except FileOpeningException as ex:
        self.ui.show_message(ex.message, "Ошибка",
QMessageBox.Critical)
    except ModelSettingException as ex:
        self.ui.show_message(ex.message, "Ошибка",
QMessageBox.Critical)
    else:
        self.hide()

    try:

self.ui.map_view.add_raster_layer(self.elements["layer_name"].text(),

"../data/result/result_" + str(self.elements["end"]

.dateTime().toString(

"yyyy_MM_dd_HH_mm")) + ".tif",

(self.elements["south_border"].value(),

self.elements["west_border"].value()),

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

(self.elements["north_border"].value()),

self.elements["east_border"].value()),

color=gdal.GCI_RedBand)
        self.ui.update_layers_list()
    except FileOpeningException as ex:
        self.ui.show_message(ex.message, "Ошибка",
QMessageBox.Critical, self.element)

    def hide(self):
        self.element.hide()

    def check_file(self, file_name):
        ds = gdal.Open(self.elements["relief_path"].text(),
gdal.GA_ReadOnly)
        if not ds:
            raise FileOpeningException("Ошибка при открытии файла" +
file_name + "!")
        else:
            ds = None

    def check_model_params(self):
        self.check_file(self.elements["relief_path"].text())
        self.check_file(self.elements["fuel_path"].text())

        if not path.exists(self.elements["fuel_codes_path"].text()):
            raise FileOpeningException("Файла кодов топлива по
данному пути не существует")

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

    if not path.exists(self.elements["weather_path"].text()):
        raise FileOpeningException("Файла погоды по данному пути
не существует")

    if not path.exists(self.elements["ignition_path"].text()):
        raise FileOpeningException("Файла территории начального
возгорания"

                                " по данному пути не
существует")

    # Даты не должны заходить одна за другую.
    if
self.elements["start"].dateTime().secsTo(self.elements["end"].dateTim
e()) <= 0:
        raise ModelSettingException("Дата окончания должна быть
позже даты начала")

    # Расположение координат.
    if self.elements["south_border"].value() >
self.elements["north_border"].value():
        raise ModelSettingException("Южная граница не может быть
выше северной!")
    if self.elements["west_border"].value() >
self.elements["east_border"].value():
        raise ModelSettingException("Западная граница не может
быть левее восточной!")

    if not
self.ui.map_view.check_layer_name(self.elements["layer_name"].text())
:

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
        raise LayerAddingException("Некорректное название слоя!")

    if
self.ui.map_view.has_layer(self.elements["layer_name"].text()):
        raise LayerAddingException("Слой с таким названием уже
существует")
    return 0
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.3 Файл MapView.py

```

import os
import geojson
from osgeo import gdal

from Components.Exceptions import LayerAddingException,
FileOpeningException, MapCreatingException, \
    LayerNotFoundException
from Components.Layers import RasterLayer, VectorLayer
from Components.MapEngine import DEFAULT_HTML,
OSM_TILE_CREATION_SCRIPT, \
    MAP_CREATION_SCRIPT, ADD_TILE_TO_MAP_SCRIPT,
RASTER_LAYER_CREATION_SCRIPT, GEOJSON_LAYER_CREATION_SCRIPT, \
    GEOJSON_LAYER_ADD_DATA_SCRIPT, SHOW_LAYER_SCRIPT,
HIDE_LAYER_SCRIPT, \
    REMOVE_LAYER_SCRIPT
from Components.Utilities import image_to_data, shp_to_json

class MapView:
    TILES_STRING_TO_SCRIPT = {"OpenStreetMap":
OSM_TILE_CREATION_SCRIPT}

    def __init__(self, window, map_tiles="OpenStreetMap",
save_file_path=None, ui=None):
        if map_tiles not in MapView.TILES_STRING_TO_SCRIPT.keys(): #
["OpenStreetMap", "Mapbox Bright", "Mapbox Control Room", "Stamen"]:
            raise Exception("Undefined map tiles")
        self.layers = []
        self.save_file_path = save_file_path

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

self.ui = ui
self.map_tiles = map_tiles
self.window = window
self.window.setHtml(DEFAULT_HTML)
self.window.loadFinished.connect(self.on_load_finished)

def on_load_finished(self, status):
    if status:
        if self.save_file_path is None:
            self.window.page().runJavaScript(MAP_CREATION_SCRIPT
+ MapView.TILES_STRING_TO_SCRIPT[self.map_tiles] +

ADD_TILE_TO_MAP_SCRIPT)

def has_layer(self, layer_name, return_layer=False):
    index = -1
    for i in range(0, len(self.layers)):
        if self.layers[i].name == layer_name:
            index = i
    if return_layer:
        if index == -1:
            return None
        else:
            return self.layers[index]
    else:
        return index != -1

def add_map_layer(self, layer_name, map_type):
    if self.has_layer(layer_name):
        raise LayerAddingException("Слой с таким названием уже

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

существует!")

```
def add_raster_layer(self, layer_name, file_path,
upper_left_bound, lower_right_bound,
color=gdal.GCI_GrayIndex, data=None):
    if not self.check_layer_name(layer_name):
        raise LayerAddingException("Некорректное название слоя!")
    if self.has_layer(layer_name):
        raise LayerAddingException("Слой с таким названием уже
существует")

    if lower_right_bound[0] < upper_left_bound[0]:
        raise LayerAddingException("Южная граница не может быть
выше северной!")
    if lower_right_bound[1] < upper_left_bound[1]:
        raise LayerAddingException("Западная граница не может
быть левее восточной!")

    if data is None and not os.path.exists(file_path):
        raise FileOpeningException("Файл не найден!")
    else:
        bounds = [upper_left_bound, lower_right_bound]
        string_bounds = "[" + str(bounds[0][0]) + ", " +
str(bounds[0][1]) + "], [" + \
str(bounds[1][0]) + ", " +
str(bounds[1][1]) + "]"
        if data is None:
            data = image_to_data(file_path, color)
        self.window.page().runJavaScript("var createLayerData =
'" + data + "';\n" +
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
RASTER_LAYER_CREATION_SCRIPT % (layer_name, string_bounds,  
layer_name))  
  
        self.layers.append(RasterLayer(layer_name, data, bounds))  
  
def add_vector_layer(self, layer_name, path, data=None):  
    global geo_data  
    if not self.check_layer_name(layer_name):  
        raise LayerAddingException("Некорректное название слоя")  
    if self.has_layer(layer_name):  
        raise LayerAddingException("Слой с таким названием уже  
существует!")  
    if not os.path.exists(path):  
        raise FileOpeningException("Файл не найден!")  
  
    if data is None:  
        try:  
            file_format = os.path.splitext(path)[-1][1:]  
  
            if file_format == ".shp":  
                path = shp_to_json(path)  
  
            geo_file = open(path, 'r')  
            geo_data = geojson.load(geo_file)  
            geo_file.close()  
        except Exception:  
            raise FileOpeningException("Невозможно прочитать  
файл!")  
    else:  
        self.layers.append(VectorLayer(layer_name,
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
str(geo_data)))

self.window.page().runJavaScript(GEOJSON_LAYER_CREATION_SCRIPT %
(layer_name, layer_name))

self.window.page().runJavaScript(GEOJSON_LAYER_ADD_DATA_SCRIPT %
(layer_name, str(geo_data)))
    else:
        self.layers.append(VectorLayer(layer_name, data))

self.window.page().runJavaScript(GEOJSON_LAYER_CREATION_SCRIPT %
(layer_name, layer_name))

self.window.page().runJavaScript(GEOJSON_LAYER_ADD_DATA_SCRIPT %
(layer_name, str(data)))

    @staticmethod
    def check_layer_name(layer_name):
        layer_name = layer_name.replace(" ", "")
        return len(layer_name) > 0

    def remove_layer(self, layer_name):
        layer = self.has_layer(layer_name, True)
        if layer is None:
            raise LayerNotFoundException("Слой не найден")
        self.layers.remove(layer)
        self.window.page().runJavaScript(REMOVE_LAYER_SCRIPT %
(layer_name, layer_name))

    def set_visible(self, layer_name, is_visible):
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
layer = self.has_layer(layer_name, True)
if layer is None:
    raise LayerNotFoundException("Слой не найден")
layer.is_visible = is_visible
if layer.is_visible:
    self.window.page().runJavaScript(SHOW_LAYER_SCRIPT %
(layer_name, layer_name))
else:
    self.window.page().runJavaScript(HIDE_LAYER_SCRIPT %
(layer_name, layer_name))
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.4 Файл MapEngine.py

```
DEFAULT_HTML = """
    <!DOCTYPE html>

    <head>

        <meta http-equiv="content-type" content="text/html; charset=UTF-8"
    />


        <script>

            L_NO_TOUCH = false;
            L_DISABLE_3D = false;
        </script>


        <script
src="https://cdn.jsdelivr.net/npm/leaflet@1.5.1/dist/leaflet.js"></scri
pt>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.min.js"
></script>
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.j
s"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/Leaflet.awesome-
markers/2.0.2/leaflet.awesome-markers.js"></script>
        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/leaflet@1.5.1/dist/leaflet.css"/>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min
.css"/>
        <link rel="stylesheet"
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-
theme.min.css"/>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.6.3/css/font-awesome.min.css"/>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/Leaflet.awesome-
markers/2.0.2/leaflet.awesome-markers.css"/>
    <link rel="stylesheet" href="https://rawcdn.githack.com/python-
visualization/folium/master/folium/templates/leaflet.awesome.rotate.css
"/>
    <style>html, body {width: 100%;height: 100%;margin: 0;padding:
0;}</style>
    <style>#map
{position:absolute;top:0;bottom:0;right:0;left:0;}</style>

    <meta name="viewport" content="width=device-width,
        initial-scale=1.0, maximum-scale=1.0, user-scalable=no"
/>
    <style>
        #main-map {
            position: relative;
            width: 100.0%;
            height: 100.0%;
            left: 0.0%;
            top: 0.0%;
        }
    </style>

</head>
<body>
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
<div class="folium-map" id="main-map"></div>

</body>
<script>
    var layers = {};
</script>
"""

MAP_CREATION_SCRIPT = """
    var mainMap = L.map("main-map",
        {
            center: [34.11, -118.5],
            crs: L.CRS.EPSG3857,
            zoom: 10,
            zoomControl: true,
            preferCanvas: false,
        }
    );
"""

OSM_TILE_CREATION_SCRIPT = """
    var mapTileLayer =
L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
    {
        "attribution": "Data by &copy; <a
href='http://openstreetmap.org'>OpenStreetMap</a>, under <a
href='http://www.openstreetmap.org/copyright'>ODbL</a>.",
        "detectRetina": false,
        "maxNativeZoom": 18,
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```

        "maxZoom": 18,
        "minZoom": 0,
        "noWrap": false,
        "opacity": 1,
        "subdomains": "abc",
        "tms": false
    }
);
""

ADD_TILE_TO_MAP_SCRIPT = ""
    mapTileLayer.addTo(mainMap);
""

GEOJSON_LAYER_CREATION_SCRIPT = ""
    layers["%s"] = L.geoJson(null, { onEachFeature: (feature, layer) =>
{ layer.on({ click: function(e) {
mainMap.fitBounds(e.target.getBounds()); }); }); });
    layers["%s"].addTo(mainMap);
""

GEOJSON_LAYER_ADD_DATA_SCRIPT = ""
    layers["%s"].addData(%s);
""

RASTER_LAYER_CREATION_SCRIPT = ""
    layers["%s"] = L.imageOverlay(createLayerData, %s);
    layers["%s"].addTo(mainMap);
""

REMOVE_LAYER_SCRIPT = ""

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
mainMap.removeLayer(layers["%s"]);
delete layers["%s"];
""

SHOW_LAYER_SCRIPT = ""
    if (!mainMap.hasLayer(layers["%s"])) {
        layers["%s"].addTo(mainMap);
    }
""

HIDE_LAYER_SCRIPT = ""
    if (mainMap.hasLayer(layers["%s"])) {
        mainMap.removeLayer(layers["%s"]);
    }
""
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.5 Файл Layers.py

```
from Components.Exceptions import LayerCreatingException

class Layer:
    def __init__(self, name, layer_type):
        if layer_type not in ['raster', 'vector']:
            raise LayerCreatingException("undefined type of layer")
        self.name = name
        self.type = layer_type
        self.is_visible = True

class RasterLayer(Layer):
    def __init__(self, name, data, bounds):
        super().__init__(name, "raster")
        self.data = data
        self.bounds = bounds

class VectorLayer(Layer):
    def __init__(self, name, data):
        super().__init__(name, "vector")
        self.data = data
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.6 Файл Utilities.py

```

from Components.Exceptions import LayerAddingException

_VALID_URLS = set(uses_relative + uses_netloc + uses_params)
_VALID_URLS.discard('')

def _is_url(url):
    try:
        return urlparse(url).scheme in _VALID_URLS
    except Exception:
        return False

def write_png(data, color = gdal.GCI_GrayIndex):
    ds = gdal.Open(data)

    if ds.RasterCount > 1:
        raise LayerAddingException("Растровый .tif файл не должен "
                                    "содержать более 1 слоя!")

    band = ds.GetRasterBand(1)

    root_ext = os.path.splitext(data)[0]
    output_path = os.path.basename(root_ext) + ".png"

    band.SetRasterColorInterpretation(color)
    band.SetNoDataValue(0)
    stats = band.GetStatistics(True, True)

    gdal.Translate(output_path, ds, format="PNG",

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
        scaleParams=[[min(stats), max(stats), [0, 255]]])

del band
del ds
return output_path


def image_to_data(path, color = gdal.GCI_GrayIndex):
    if isinstance(path, str) and not _is_url(path):
        file_format = os.path.splitext(path)[-1][1:]

        if file_format == "tif":
            png_path = write_png(path, color)
            with io.open(png_path, 'rb') as f:
                img = f.read()
                b64encoded = base64.b64encode(img).decode('utf-8')
                url = 'data:image/png;base64,{}'.format(b64encoded)
        else:
            with io.open(path, 'rb') as f:
                img = f.read()
                b64encoded = base64.b64encode(img).decode('utf-8')
                url = 'data:image/{};base64,{}'.format(file_format,
b64encoded)

            return url.replace('\n', ' ')
        else:
            url = json.loads(json.dumps(path))
            return url.replace('\n', ' ')


def shp_to_json(path):
    root_ext = os.path.splitext(path)[0]
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```

output_path = os.path.basename(root_ext) + ".geojson"

reader = shapefile.Reader(path)
fields = reader.fields[1:]
field_names = [field[0] for field in fields]
buffer = []
for sr in reader.shapeRecords():
    atr = dict(zip(field_names, sr.record))
    geom = sr.shape.__geo_interface__
    buffer.append(dict(type="Feature",
                       geometry=geom, properties=atr))

geojson = open(output_path, "w")
try:
    geojson.write(dumps({"type": "FeatureCollection", "features":
buffer}, indent=2, default=str) + "\n")
except TypeError:
    print("Hi!")
geojson.close()
return output_path

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.7 Файл Layers.py

```
from Components.Exceptions import LayerCreatingException
```

```
class Layer:
```

```
    def __init__(self, name, layer_type):
        if layer_type not in ['raster', 'vector']:
            raise LayerCreatingException("undefined type of layer")
        self.name = name
        self.type = layer_type
        self.is_visible = True
```

```
class RasterLayer(Layer):
```

```
    def __init__(self, name, data, bounds):
        super().__init__(name, "raster")
        self.data = data
        self.bounds = bounds
```

```
class VectorLayer(Layer):
```

```
    def __init__(self, name, data):
        super().__init__(name, "vector")
        self.data = data
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.8 Файл Exceptions.py

```
class LayerAddingException(Exception):
```

```
    def __init__(self, message):  
        super().__init__(message)  
        self.message = message
```

```
class LayerCreatingException(Exception):
```

```
    def __init__(self, message):  
        super().__init__(message)  
        self.message = message
```

```
class MapCreatingException(Exception):
```

```
    def __init__(self, message):  
        super().__init__(message)  
        self.message = message
```

```
class FileOpeningException(Exception):
```

```
    def __init__(self, message):  
        super().__init__(message)  
        self.message = message
```

```
class NotVectorLayerException(Exception):
```

```
    def __init__(self, message):  
        super().__init__(message)  
        self.message = message
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
class LayerNotFoundException(Exception):  
    def __init__(self, message):  
        super().__init__(message)  
        self.message = message
```

```
class ModelSettingException(Exception):  
    def __init__(self, message):  
        super().__init__(message)  
        self.message = message
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.06-01 12 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

[illegible]