МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра програмного забезпечення комп’ютерних систем

**IC реєстрації та контролю повідомлень учасників форуму**

(літня технологічна практика)

**«УЗГОДЖЕНО»**

Керівник проекту

\_\_\_\_\_\_\_Комісарчук В.В.

(підпис)

Виконав студент 443ск групи

\_\_\_\_\_\_\_ Корольов М.С

(підпис)

Екзаменаційна оцінка з технологічної практики:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № пп | Прізвище, ініціали студента | Підсумкова оц. за шкалою ВНЗ | Підсумкова оц. за шкалою ECTS | Підсумкова оц. за націонал. шкалою | Підпис керівника практики |
| 1. |  |  |  |  |  |
| 2. |  |  |  |  |  |
| 3. |  |  |  |  |  |

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_ р.

Чернівці, 2019

**ЗМІСТ**

1. ТЕХНІЧНЕ ЗАВДАННЯ…………………………………………………….....4

1.1. Призначення розробки………………………………………………........4

1.1.1. Тема технологічної практики……………………………………….4

1.1.2. Основні функціональні можливості……………………………......4

1.2. Аналіз вимог до програмного забезпечення…………………………….4

1.2.1. Функціональні вимоги........................................................................4

1.2.2. Вимоги до складу та параметрів технічних засобів……………….5

1.2.3. Вимоги до інтерфейсу користувача………………………………...5

1.2.4. Вимоги до інформаційної та програмної сумісності………….......5

2. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ…………………...6

2.1. Опис функціональних можливостей…………………………………...6

2.2. Опис та обґрунтування обраної архітектури…………………………..6

2.3. Опис діаграми модулів………………………………………………..6

3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ………………..8

3.1. Опис і обґрунтування обраних програмних засобів…………………..8

3.2. Опис графічного інтерфейсу користувача……………………….…….8

ВИСНОВКИ……………………………………………………………………...14

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………….15

ДОДАТКИ………………………………………………………………………..16

Додаток А………………………………………………………………….16

Додаток B………………………………………………………………….18

Додаток C………………………………………………………………….20

Додаток D………………………………………………………………….21

Додаток E………………………………………………………………….22

Додаток F………………………………………………………………….23

Додаток G………………………………………………………………….25

Додаток H………………………………………………………………….26

Додаток I………………………………………………………………….27

Додаток J………………………………………………………………….28

Додаток K………………………………………………………………….29

Додаток L………………………………………………………………….30

Додаток M…………………………………………...…………………….31

Додаток N………………………………………………………………….32

Додаток O………………………………………………………………….33

Додаток P………………………………………………………………….34

Додаток Q………………………………………………………………….35

Додаток S…………………………………………………………………..36

**1. ТЕХНІЧНЕ ЗАВДАННЯ**

**1.1 Призначення розробки**

**1.1.1 Тема технологічної практики**

Розробка програмного забезпечення виконується на підставі рішення засідання кафедри ПЗКС про затвердження тем на технологічну практику.

Назва отриманої теми «ІС система реєстрації та контролю повідомлень учасників форума».

**1.1.2 Основні функціональні можливості**

Розроблене програмне забезпечення повинно забезпечити реалiзацiю функiй веб-форума та можливiсть контролювати данi ресурса.

**1.2 Аналіз вимог до програмного забезпечення**

**1.2.1 Функціональні вимоги**

Програмне забезпечення, що було розроблене в ході виконання технологічної практики, повинно забезпечувати наступні функціональні вимоги:

1. Працювати в будь-якому браузері;
2. Забезпечити контроль повідомлень;
3. ведення логу роботи програми;
4. збереження вхідних та вихідних даних.

**1.2.2 Вимоги до складу та параметрів технічних засобів**

Програма повинна працювати на комп’ютерах, де стабільно працює операційна система Windows XP. Інших вимог до апаратної частини не висувається.

**1.2.3 Вимоги до інтерфейсу користувача**

Програма повинна мати інтуїтивно зрозумілий, зручний та візуально-приємний інтерфейс.

**1.2.4 Вимоги до інформаційної та програмної сумісності**

Програма повинна працювати на комп’ютерах, де встановлена операційна система Windows XP (SP1, SP2, SP3, Seven).

**2. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ**

**2.1 Опис функціональних можливостей**

Програмне забезпечення, що було розроблене в процесі написання технологічної практики, забезпечує наступні функціональні можливості, що реалізують поставлене завдання:

1. Введення логу роботи програми.
2. Контроль вхідних та вихідних повідомлень.
3. Створення тем для повідомлень.

**2.2 Опис та обґрунтування обраної архітектури**

У розробленому програмному забезпеченні користувач має змогу виконувати 6 основних процесiв в залежності від мети використання програми. Це може бути створення тем, створення повiдомлень, аутентифiкацiя користувачiв, редагування категорiй, тем та повiдомлень.

**2.3 Опис діаграми модулів**

Робота розробленого програмного продукту реалізується наступними файлами модулів:

1. Модуль express.
2. Модуль mongoose.
3. Модуль crypto.
4. Модуль cookie-parser.
5. Модуль express-session.
6. Модуль pug
7. Модуль Controllers

8. Модуль Models

9. Модуль Routers

Модуль express є головним модулем розробленого програмного забезпечення. В ньому містяться функції, якi керують поведiнку всю систему.

Модуль mongoose забезпечує взаемодiю з базою даних MongoDB.

Модуль crypto вiдповiдає за шифрування паролiв користувачiв.

Модуль cookie-parser створює кукi користувачiв.

Модуль express-session необхiдний для створення сесiй користувачiв.

Модуль pug виконує роль шаблонiзатора HTML.

**3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**3.1 Опис і обґрунтування обраних програмних засобів**

Розробка програмного забезпечення здійснена за допомогою середовища NodeJS.

NodeJS – зручний засіб для швидкої розробки на сервернiй платформi.

**3.2. Опис графічного інтерфейсу користувача**

Розроблене програмне забезпечення – набір модулів, які об’єднані графічним інтерфейсом користувача.

Головна сторiнка складається зi списку доступних категорiй та навiгацiйної панелi на якiй знаходиться, також, форма для авторизацiї користувача.

На рис.3.2.1 зображена головна сторiнка розробленого продукту:

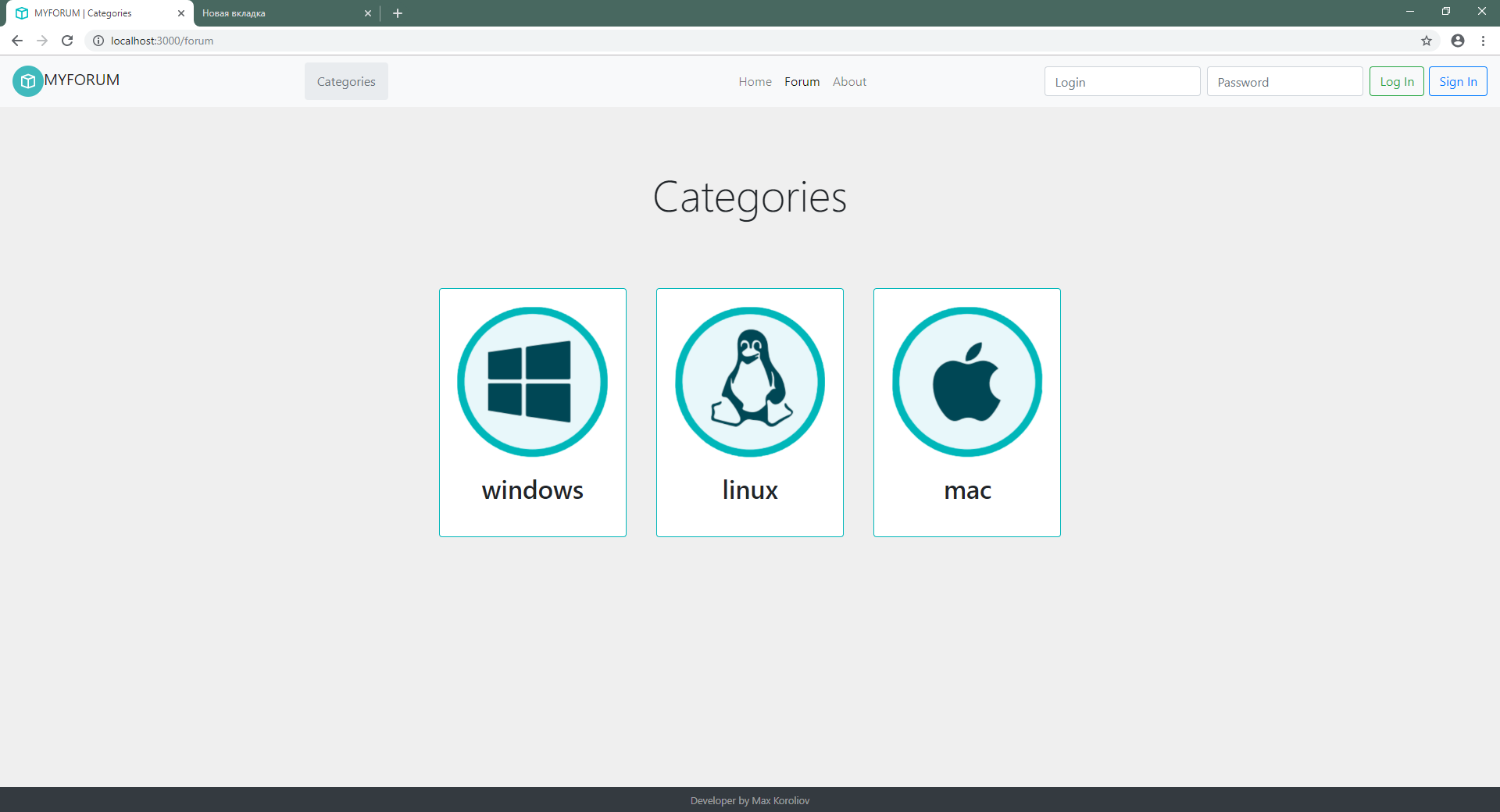


Рисунок 3.2.1 – Головна сторiнка програмного продукту

Рядок навiгацiйної панелi складається з елементiв логотипу, вказiвника на мiсцеположення поточної сторiнки в навiгацiйнiй iєрархiї, ссилок на роздiли продукту, та форми авторизацiї.

Якщо користувач уже авторизований, тодi форма замiниться на ссилку в особистий кабiнет користувача.

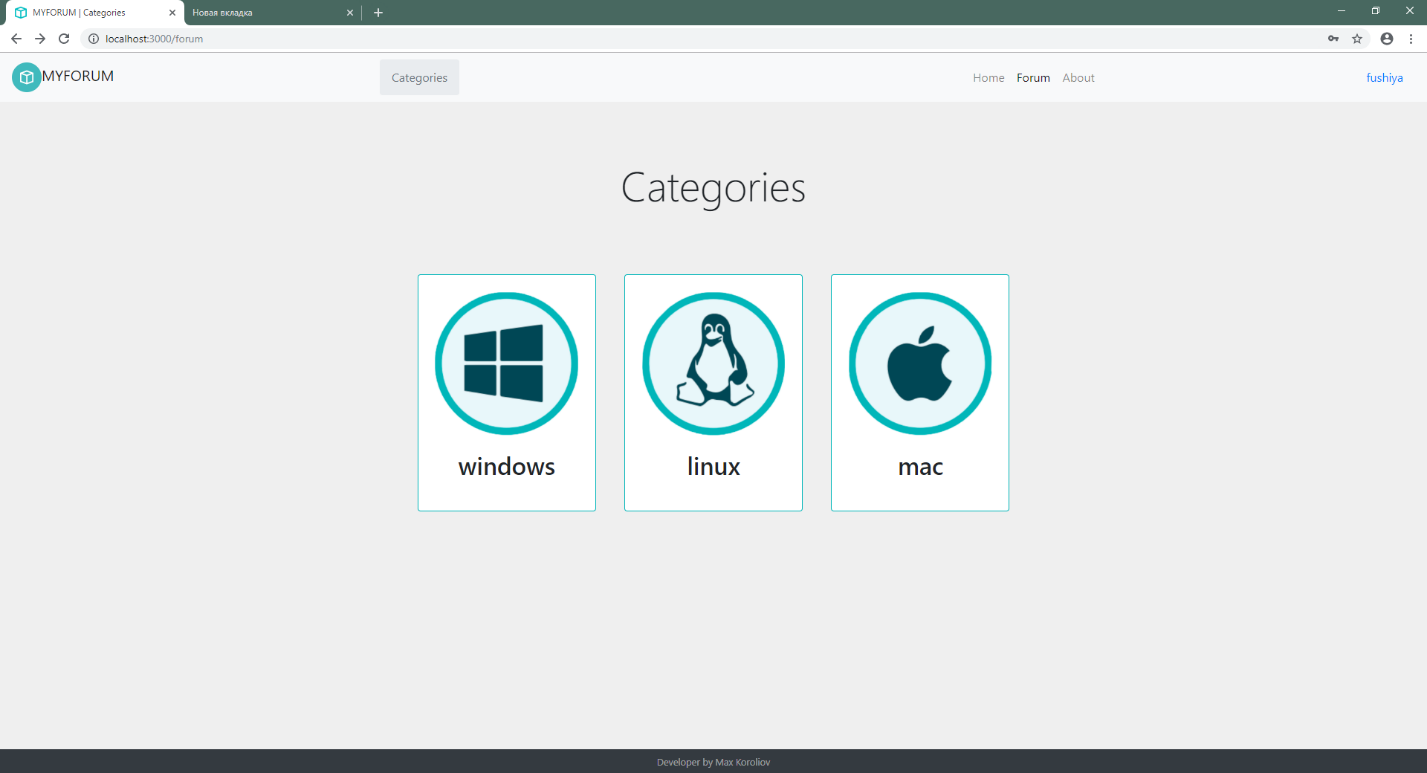


Рисунок 3.2.2 – Головна сторiнка програмного продукту з авторизованим користувачем

При натисканнi на тему, вiдкриється список доступних тем даної категорiї.

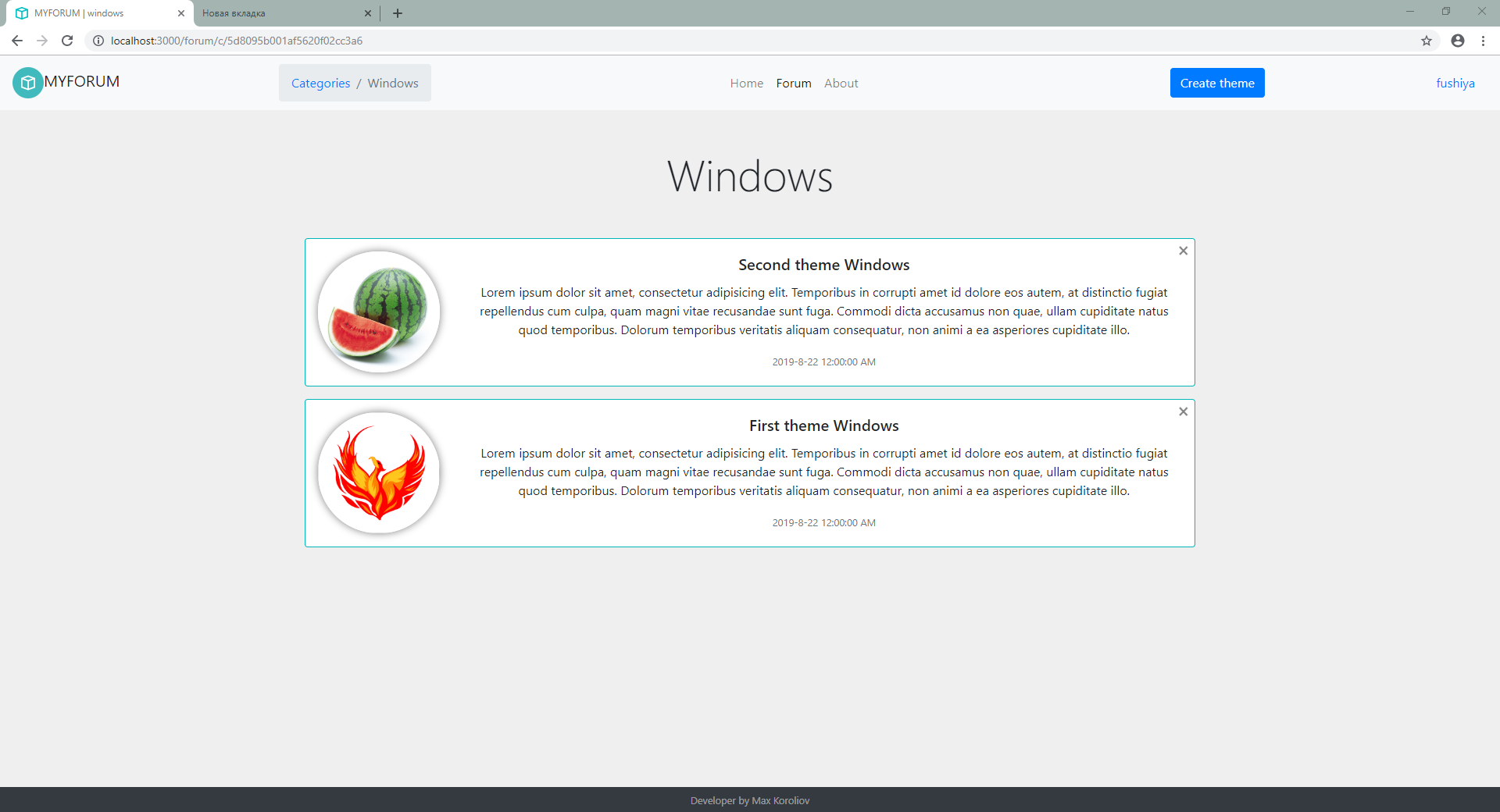


Рисунок 3.2.3 – Сторiнка зi списком тем

На сторiнцi тем, у навiгацiйнiй панелi з'явився новий елемент – кнопка «Створити тему». Створювати тему можуть тiльки авторизованi користувачi, в iншому випадку здiсниться перенаправлення на головну сторiнку.

Список тем сортуєтся вiд найновiшого, до найранiшого створеного. Переконатись у цьому можна, глянувши пiд опис теми – там вiдображається дата створення теми.

Обравши тему та на неї натиснувши, сторiнка вiдобразить список усiх повiдомлень даної теми, сортуючи вiд найпершого повiдомлення до найновiшого.

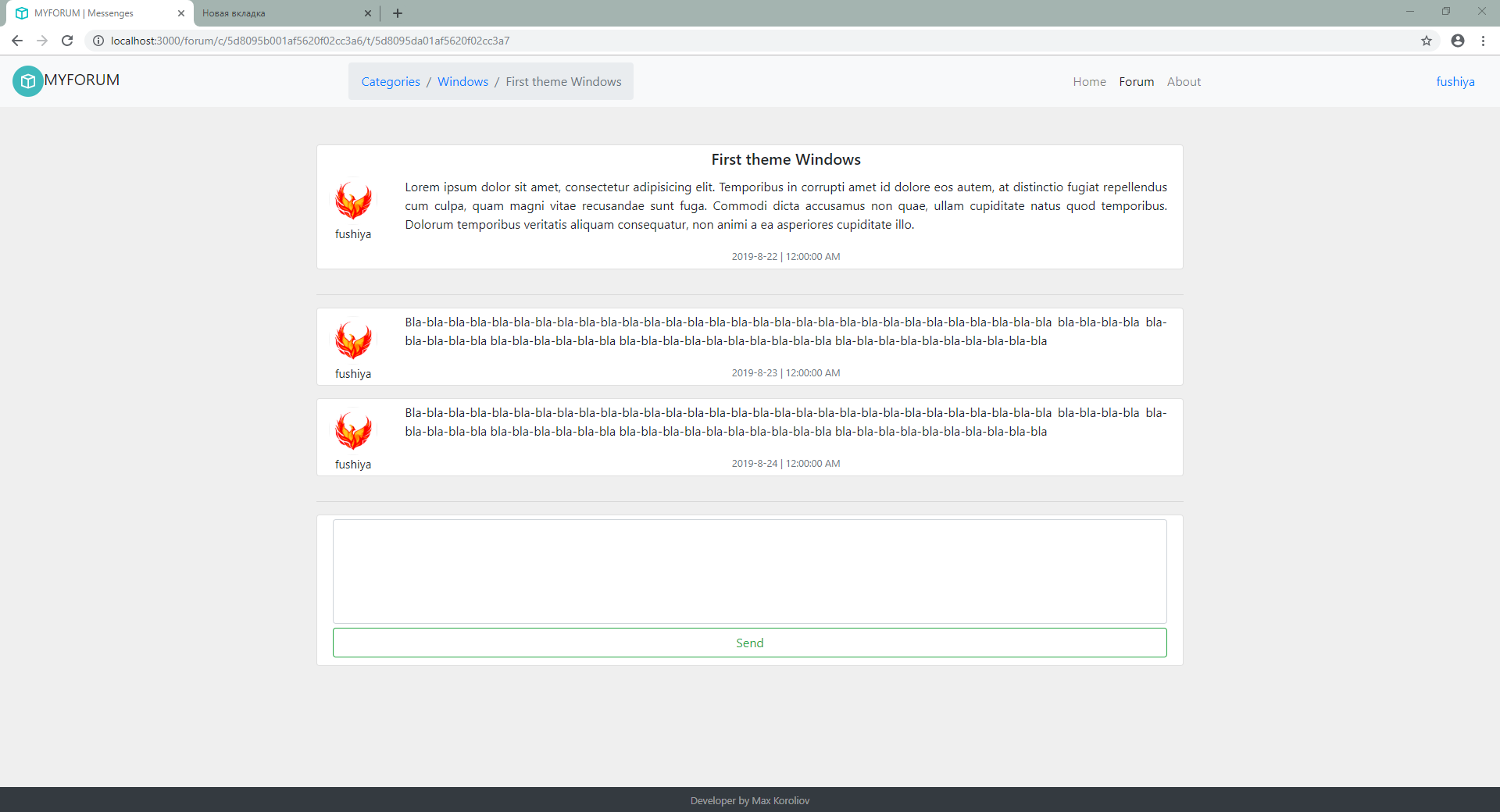


Рисунок 3.2.4 – Сторiнка повiдомлень теми.

Якщо користувач не авторизований, то вiн не зможе вiдправити повiдомлення.

Якщо натиснути на логiн, який знаходиться на навiгацiйнiй панелi, то вiдкриється особистий кабiнет користувача, у якому можна редагувати данi користувача.

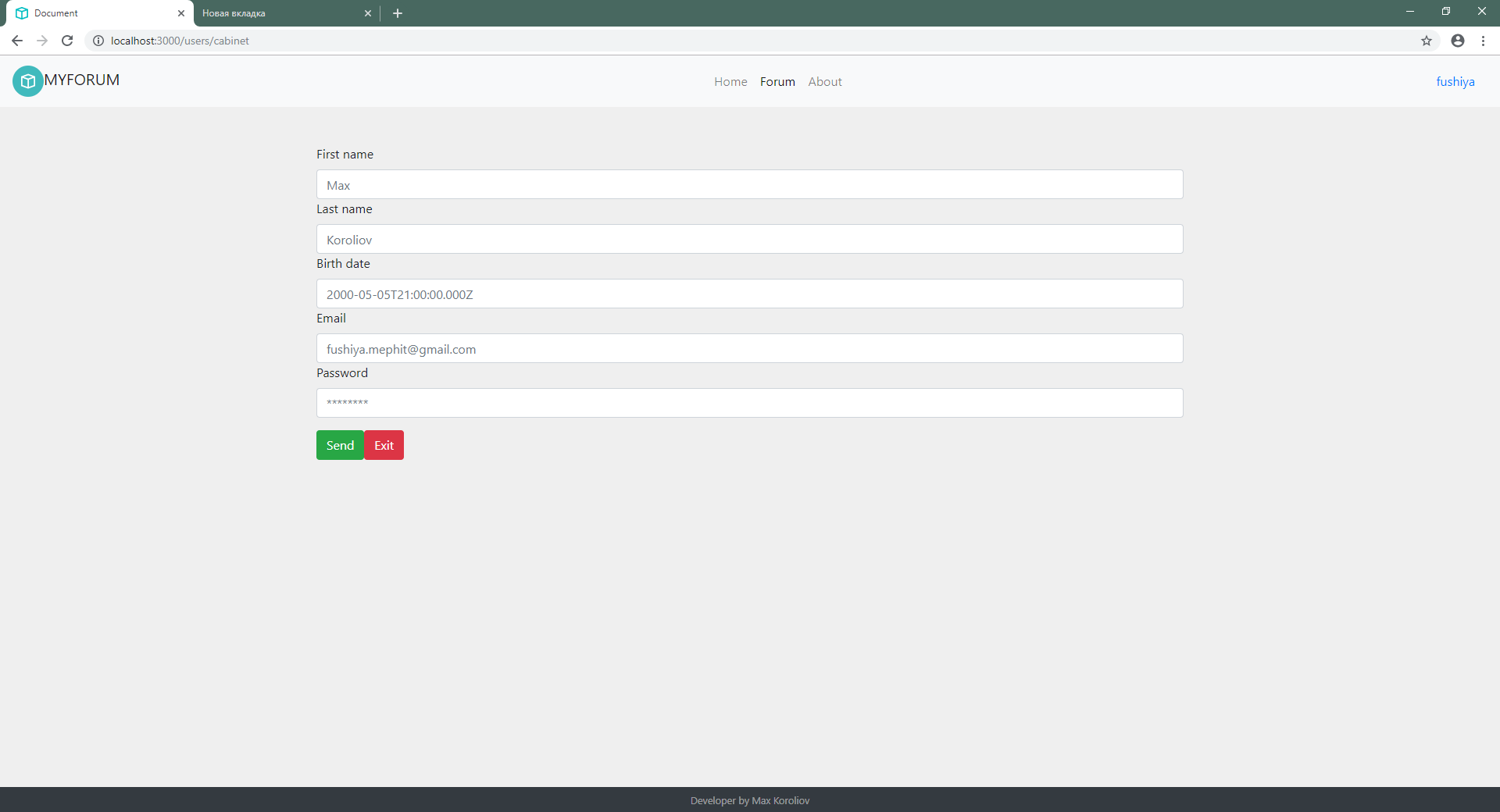


Рисунок 3.2.5 – Особистий кабiнет користувача

Для того, щоб створити категорiю, необхiдно перейти на адмiнiнiстративну панель. Щоб потрапити на адмiнiнiстративну панель, необхiдно перейти на сторiнку /admin та бути авторизованим користувачем з правами адмiнiстратора. В iншому випадку здiйсниться перенаправлення на головну сторiнку. У базi даних на даний момент зареєстровано всього три користувача - права адмiнiстратора має тiльки користувач з логiном “fushiya”.

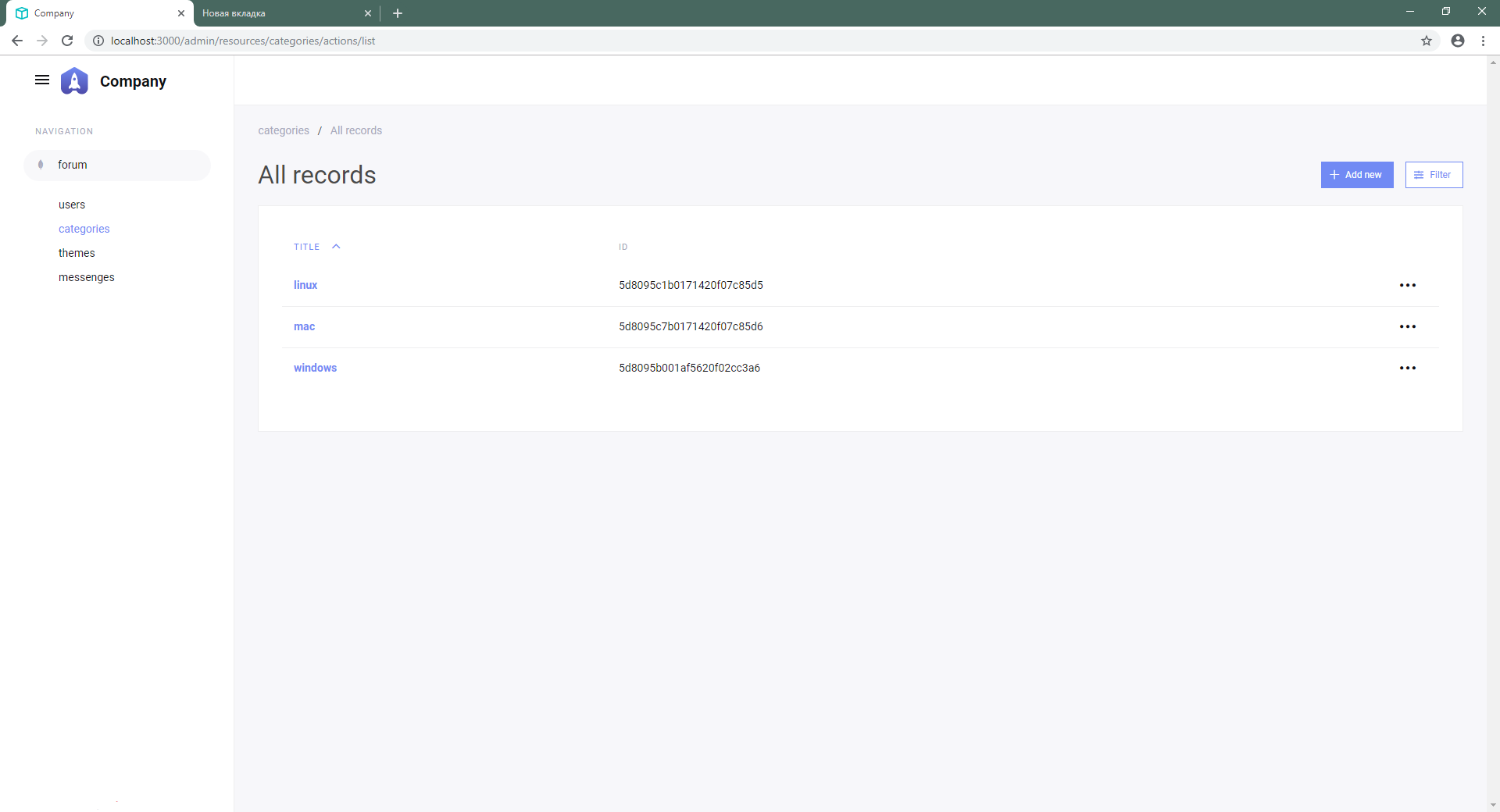


Рисунок 3.2.6 – Панель адмiнiстратора

На панелi адмiнiстратора, також є можливiсть редагувати та контролювати будь-яку iнформацiю на даному ресурсi.

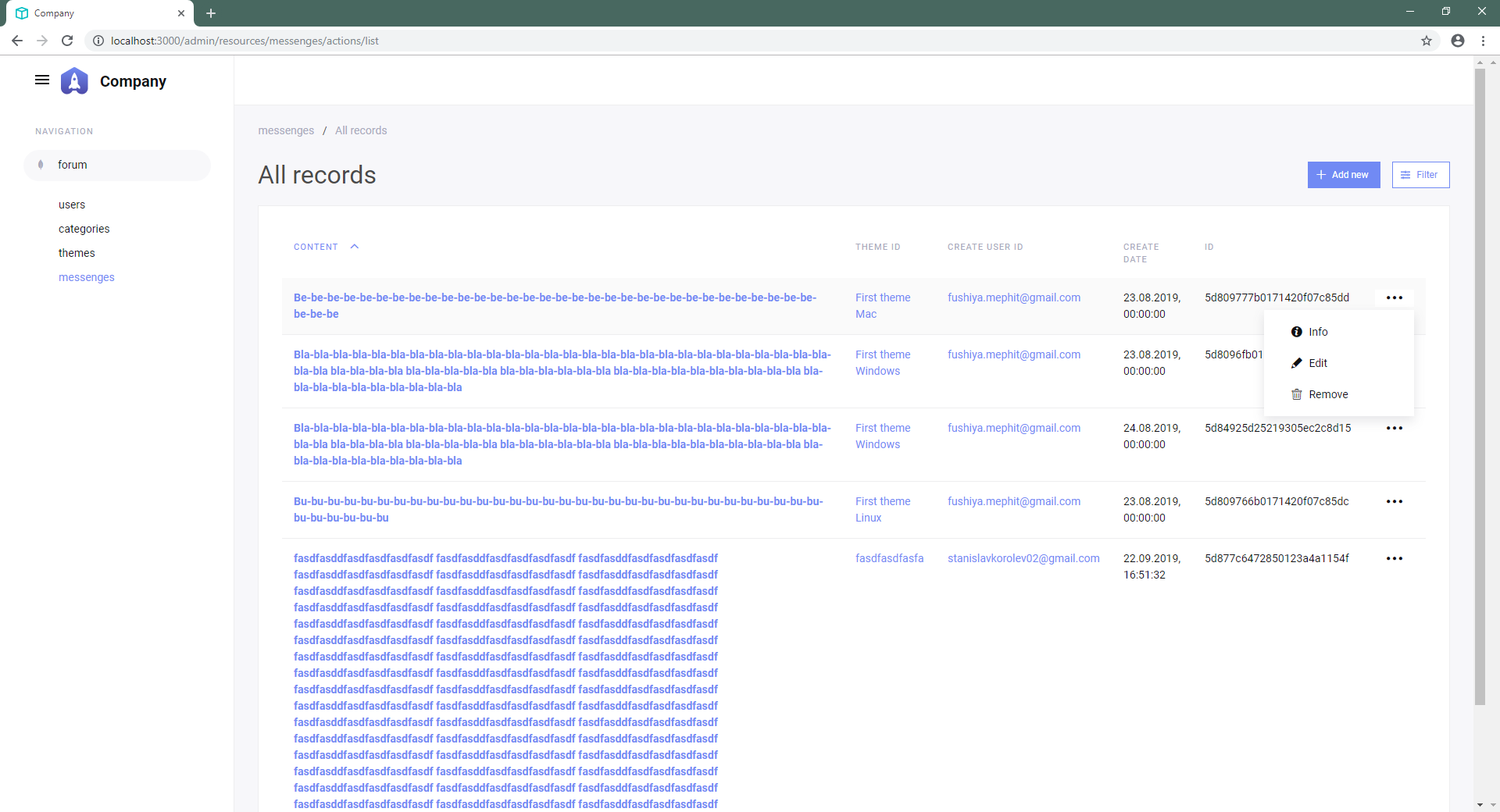


Рисунок 3.2.7 – Редагування повiдомлень учасникiв форума

**ВИСНОВКИ**

В процесі виконання технологічної практики було створено програмне забезпечення, що реалізує веб-форум та його контроль.

Розроблена система забезпечуює виконання функцiй веб-форума та здатнiсть контролювати систему за допомогою адмiнiстративної панелi.

Розроблений інтуїтивно-зрозумілий інтерфейс із зручною панеллю керування.

Проведено тестування та корекцію роботи розробленого програмного продукту.

Складено необхідну технічну документацію, що повинна постачатись разом з розробленим програмним забезпеченням.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Ватолин Д.С., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2002.
2. Романов В.Ю. Популярные форматы файлов для хранения графических изображений на IBM PC.– М.:Унитех, 1992.
3. Семенюк В. В. Экономное кодирование дискретной информации. – СПб.: СПбГИТМО (ТУ), 2001.
4. Ватолин Д., Ратушняк А., Смирнов М. Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.ДИАЛОГ-МИФИ, 2003. – 384с.
5. Метод LZW-сжатия данных [Електронний ресурс]. – Режим доступу до документа: <http://algolist.manual.ru/compress/standard/lzw.php>.

**ДОДАТКИ**

**Додаток А. server.js**

const express = require('express');

const session = require('express-session');

var cookieParser = require('cookie-parser');

const mongodb = require('mongodb');

var mongoStore = require('connect-mongo')(session);

const mongoose = require('mongoose');

const multer = require('multer');

const pug = require('pug');

const app = express();

var port = process.env.PORT || 3000;

const Routers = require('./modules/routers/all\_routers');

const host = process.env.HOST || 'localhost';

const AdminBro = require('admin-bro');

const AdminBroExpress = require('admin-bro-expressjs');

const AdminBroMongoose = require('admin-bro-mongoose');

const Controllers = require('./modules/controllers/all\_controllers');

const Models = require('./modules/models/all\_models');

AdminBro.registerAdapter(AdminBroMongoose);

app.set('views engine', 'pug');

app.use(express.static(\_\_dirname + "/public"));

const db = "mongodb://localhost:27017/forum";

const adminBro = new AdminBro({

    databases: [],

    rootPath: '/admin',

    resources: [Models.users, Models.categories, Models.themes, Models.messenges],

});

const adminRouter = AdminBroExpress.buildRouter(adminBro);

mongoose.connect(db, { useNewUrlParser: true }, function(err){

    if(err) return console.log(err);

    app.listen(port, function(){

        console.log("OK...");

    });

});

app.use(cookieParser());

app.use(session({

    secret: 'summer\_practic\_job',

    resave: false,

    saveUninitialized: false,

    store: new mongoStore({

      url: db,

    })

}));

app.use(adminBro.options.rootPath, Controllers.users.checkAdmin, adminRouter);

app.use('/users', Routers.users);

app.use('/post', Routers.post);

app.use('/forum', Routers.forum);

app.use('/', Routers.index);

**Додаток B. /modules/controllers/usersController.js**

const express = require('express');

const Models = require('../models/all\_models');

const hash = require('../adapterPass');

exports.auth = (req, res) => {

    if (!req.body) return res.status(400);

        Models.users.findOne({login: req.body.ulogin, pass: hash(req.body.upass, 'secret')}, (er, usr) => {

            req.session.auth = true;

            req.session.user = usr;

            res.send(usr);

        });

}

exports.cabinet = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    Models.users.findOne({\_id: ses.\_id}, (er, usr) => {

        res.render('cabinet.pug', {

            usr: usr,

            sdata: ses

        });

    });

}

exports.signout = (req, res) => {

    req.session.destroy((er) => {

        if (er) return console.log(er);

        res.redirect('/forum');

    });

}

exports.putUsers = (req, res) => {

    if (!req.body) return res.send(400);

    const upUser = req.body;

    console.log(Object.keys(req));

    console.log(req.files);

    console.log(req.file);

    Models.users.findOne({\_id: req.session.user.\_id}, (er, nusr) => {

        res.send([nusr, upUser]);

    });

}

exports.signon = (req, res) => {

    res.render('signon.pug');

}

exports.register = (req, res) => {

    if (!req.body) return res.send(400);

    const hashpass = hash(req.body.newUsr.pass, 'secret');

    const newUser = new Models.users({

        first\_name: req.body.newUsr.first\_name,

        last\_name: req.body.newUsr.last\_name,

        email: req.body.newUsr.email,

        login: req.body.newUsr.login,

        pass: hashpass,

        birth\_date: req.body.newUsr.birth\_date,

        avatar: "",

        friends: [],

        register\_date: new Date(),

        role: 'participant'

    });

    newUser.save((err) => {

        if (err) return console.log(err);

        req.session.user = newUser;

        req.session.auth = true;

        res.send(true);

    });

}

exports.checkAdmin = (req, res, next) => {

    try {

        (req.session.user.role == "admin") ? next() : res.redirect('/forum');

    } catch (ex) {

        res.redirect('/forum');

    }

}

**Додаток С. /modules/controllers/categoriesController.js**

const express = require('express');

const Models = require('../models/all\_models');

exports.getCategories = (req, res) => {

    res.render('categoriesList.pug');

}

exports.setCategories = (req, res) => {

    res.send('Controllers.categories.setCategories');

}

exports.category = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    Models.categories.find({}, (er, ctg) => {

        res.render('categoriesList.pug', {

            categ: ctg,

            sdata: ses

        });

    });

}

**Додаток E. /modules/controllers/messegesController.js**

const express = require('express');

const Models = require('../models/all\_models');

exports.setMessege = (req, res) => {

    res.send('Controllers.messeges.setMessege');

}

exports.getMessege = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    const theme = req.params.theme;

    Models.messenges.find({themeId: theme}).populate([{path: 'themeId', populate: ['categoryId', 'create\_userId']}, {path: 'create\_userId'}]).exec((er, msg) => {

        if (er) return console.log(er);

        Models.themes.findOne({\_id: theme}).populate(['categoryId', 'create\_userId']).exec((err, thms) => {

            if (err) return console.log(err);

            res.render('messengesList.pug', {

                thm: thms,

                msgs: msg,

                sdata: ses

            });

        });

    });

}

exports.addMessege = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    if (!req.body) return res.send(400);

    const newMessege = new Models.messenges({

        content: req.body.content\_message,

        themeId: req.body.themeId\_message,

        create\_userId: req.session.user.\_id,

        create\_date: new Date()

    });

    newMessege.save();

    res.send([newMessege, req.session.user]);

}

exports.delMesseges = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    if (!req.body) return res.send(400);

    Models.messenges.deleteOne({name: req.body.name\_msg, content: req.body.content\_msg}, (err) => {if (err) return console.log(err);});

}

**Додаток F. /modules/controllers/themesController.js**

const express = require('express');

const Models = require('../models/all\_models');

exports.setTheme = (req, res) => {

    res.send('Controllers.themes.setTheme');

}

exports.getTheme = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    const category = req.params.category;

    Models.themes.find({categoryId: category}).populate([{path: 'categoryId'}, {path: 'create\_userId'}]).exec((er, thm) => {

        if (er) console.log(er);

        res.render('themesList.pug', {

            thms: thm.reverse(),

            sdata: ses

        });

    });

}

exports.create = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    Models.categories.findOne({\_id: req.params.category}, (er, ctgs) => {

        res.render('createTheme.pug', {

            ctg: ctgs,

            sdata: ses

        });

    });

}

exports.addTheme = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    let newTheme = new Models.themes({

        name: req.body.title\_theme,

        content: req.body.content\_theme,

        categoryId: req.body.categoryId\_theme,

        create\_userId: req.session.user.\_id,

        create\_date: new Date()

    });

    newTheme.save();

    res.send(`/forum/c/${req.body.categoryId\_theme}`);

}

exports.delThemes = (req, res) => {

    if (req.session.auth) {var ses = req.session.user} else {var ses = false;};

    if (!req.body) return res.send(400);

    console.log(req.body);

    console.log(rreq.session.user);

    Models.themes.deleteOne({name: req.body.stitle\_thm, create\_user: req.session.user.\_id}, (err) => {if (err) return console.log(err); res.send(true);});

}

**Додаток G. modules/controllers/indexController.js**

const express = require('express');

const Models = require('../models/all\_models');

exports.index = (req, res) => {

    res.redirect('/home');

}

exports.home = (req, res) => {

    res.render('home.pug');

}

**Додаток H. Modules/controllers/all\_controllers.js**

exports.index = require('./indexController');

exports.users = require('./usersController');

exports.categories = require('./categoriesController');

exports.themes = require('./themesController');

exports.messenges = require('./messegesController');

**Додаток I. modules/models/usersModel.js**

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const usersSchema = new Schema({

    first\_name: {

        type: String,

        required: true

    },

    last\_name: {

        type: String,

        required: true

    },

    birth\_date: {

        type: Date,

        required: true

    },

    email: {

        type: String,

        required: true

    },

    register\_date: {

        type: Date,

        required: true

    },

    login: {

        type: String,

        required: true,

        unique: true

    },

    pass: {

        type: String,

        required: true

    },

    avatar: {

        type: String

    },

    friends: {

        type: [String]

    },

    role: {

        type: String,

        required: true

    }

});

module.exports = mongoose.model('users', usersSchema);

**Додаток J. modules/models/all\_models.js**

exports.users = require('./usersModel');

exports.categories = require('./categoriesModel');

exports.themes = require('./themesModel');

exports.messenges = require('./messegesModel');

**Додаток К. modules/models/messegesModel.js**

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const messengesSchema = new Schema({

    content: {

        type: String,

        required: true

    },

    themeId: {

        type: Schema.Types.ObjectId,

        ref: 'themes',

        required: true

    },

    create\_userId: {

        type: Schema.Types.ObjectId,

        ref: 'users',

        required: true

    },

    create\_date: {

        type: Date,

        required: true

    }

});

module.exports = mongoose.model('messenges', messengesSchema);

**Додаток L. modules/models/themesModel.js**

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const themesSchema = new Schema({

    name: {

        type: String,

        required: true

    },

    content: {

        type: String

    },

    categoryId: {

        type: Schema.Types.ObjectId,

        ref: 'categories',

        required: true

    },

    create\_userId: {

        type: Schema.Types.ObjectId,

        ref: 'users',

        required: true

    },

    create\_date: {

        type: Date,

        required: true

    }

});

module.exports = mongoose.model('themes', themesSchema);

**Додаток М. modules/routers/forumRouter.js**

const express = require('express');

const forumRouter = express.Router();

const Controllers = require('../controllers/all\_controllers');

forumRouter.get('/c/:category/create', Controllers.themes.create);

forumRouter.get('/c/:category/t/:theme', Controllers.messenges.getMessege);

forumRouter.get('/c/:category', Controllers.themes.getTheme);

forumRouter.get('/', Controllers.categories.category);

module.exports = forumRouter;

**Додаток N. modules/routers/indexRouter.js**

const express = require('express');

const indexRouter = express.Router();

const Controllers = require('../controllers/all\_controllers');

indexRouter.get('/home', Controllers.index.home);

indexRouter.use('/', Controllers.index.index);

module.exports = indexRouter;

**Додаток O. modules/routers/postRouter.js**

const express = require('express');

const Controllers = require('../controllers/all\_controllers');

const jsonParser = express.json();

const postRouter = express.Router();

postRouter.post('/auth', jsonParser, Controllers.users.auth);

postRouter.post('/register', jsonParser, Controllers.users.register);

postRouter.post('/addtheme', jsonParser, Controllers.themes.addTheme);

postRouter.post('/addmessege', jsonParser, Controllers.messenges.addMessege);

postRouter.post('/putuser', jsonParser, Controllers.users.putUsers);

postRouter.post('/delmsg', jsonParser, Controllers.messenges.delMesseges);

postRouter.post('/delthm', jsonParser, Controllers.themes.delThemes);

module.exports = postRouter;

**Додаток P. modules/routers/usersRouter.js**

const express = require('express');

const usersRouter = express.Router();

const Controllers = require('../controllers/all\_controllers');

usersRouter.get('/cabinet', Controllers.users.cabinet);

usersRouter.get('/signout', Controllers.users.signout);

usersRouter.get('/signon', Controllers.users.signon);

module.exports = usersRouter;

**Додаток Q. modules/routers/all\_routers.js**

exports.index = require('./indexRouter');

exports.users = require('./usersRouter');

exports.forum = require('./forumRouter');

exports.post = require('./postRouter');

**Додаток R. Modules/adapterPass.js**

const crypto = require('crypto');

const hshs = [

    'sha1', 'md5', 'sha256', 'sha512'

]

module.exports = (secret, key) => {

    return crypto.createHmac(hshs[0], secret)

        .update(key)

        .digest('hex');

}

**Додаток S. modules/models/categoriesModel.js**

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const categoriesSchema = new Schema({

    title: {

        type: String,

        required: true,

        unique: true

    }

});

module.exports = mongoose.model('categories', categoriesSchema);