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

ІС 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 національ. шкалою	Підпис керівника практики
1.					
2.					
3.					

«___»_____ 201__ р.

Чернівці, 2019

ЗМІСТ

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 реалізацію функій веб-форума та можливість контролювати дані 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 процесів в залежності від мети використання програми. Це може бути створення тем, створення повідомлень, аутентифікація користувачів, редагування категорій, тем та повідомлень.

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, які керують поведінку всю систему.

Модуль mongoose забезпечує взаемодію з базою даних MongoDB.

Модуль crypto відповідає за шифрування паролів користувачів.

Модуль cookie-parser створює кукі користувачів.

Модуль express-session необхідний для створення сесій користувачів.

Модуль pug виконує роль шаблонізатора 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 серверній платформі.

3.2. Опис графічного інтерфейсу користувача

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

Головна сторінка складається зі списку доступних категорій та навігаційної панелі на якій знаходиться, також, форма для авторизації користувача.

На рис.3.2.1 зображена головна сторінка розробленого продукту:

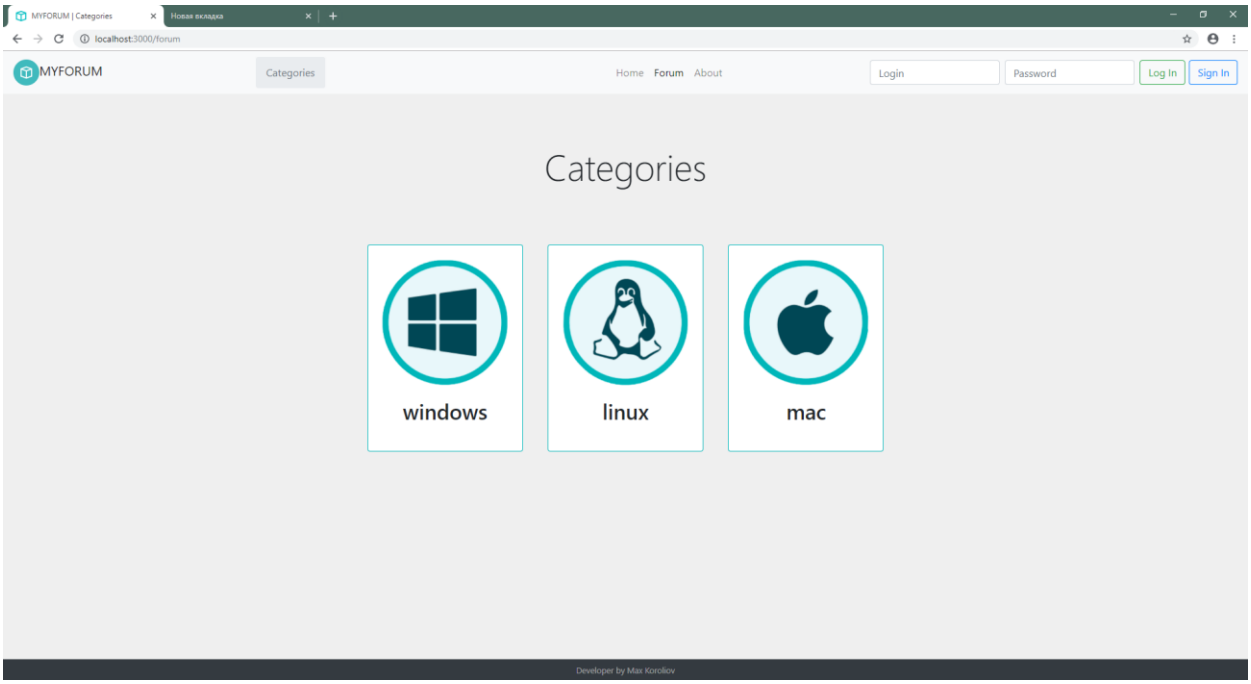

Рисунок 3.2.1 – Головна сторінка програмного продукту

Рядок навігаційної панелі складається з елементів логотипу, вказівника на місцеположення поточної сторінки в навігаційній ієрархії, ссилок на розділи продукту, та форми авторизації.

Якщо користувач уже авторизований, тоді форма заміниться на ссилку в особистий кабінет користувача.

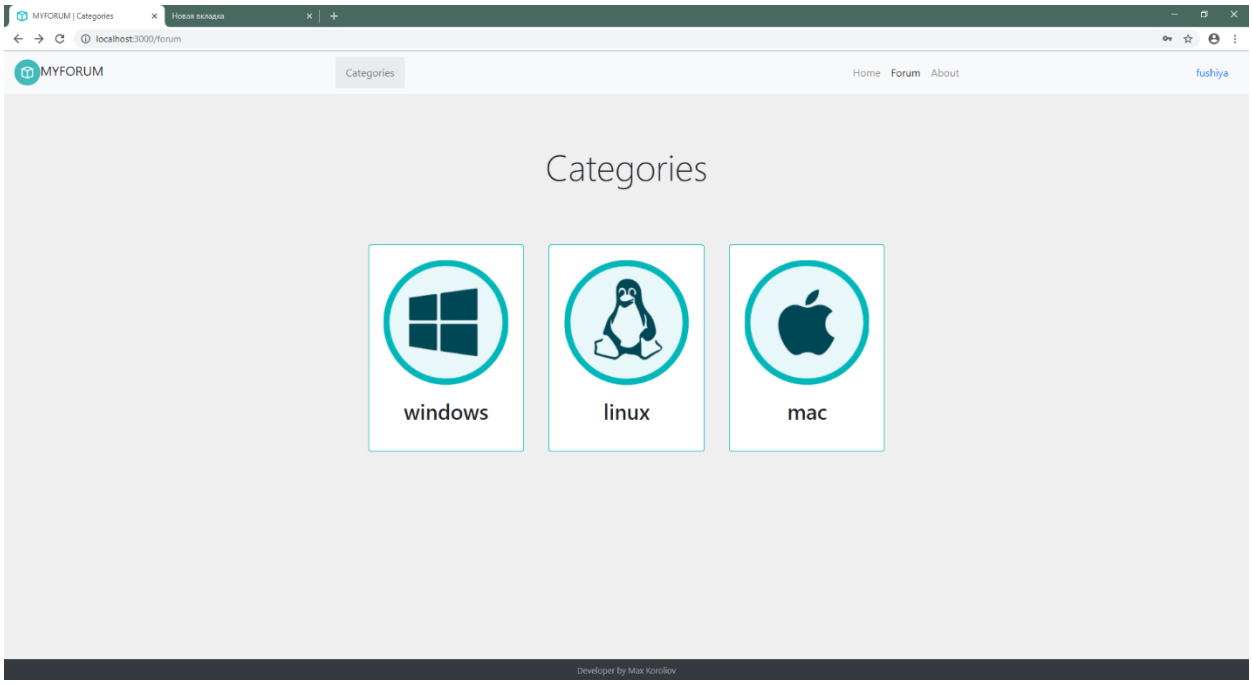

Рисунок 3.2.2 – Головна сторінка програмного продукту з авторизованим користувачем

При натисканні на тему, відкриється список доступних тем даної категорії.

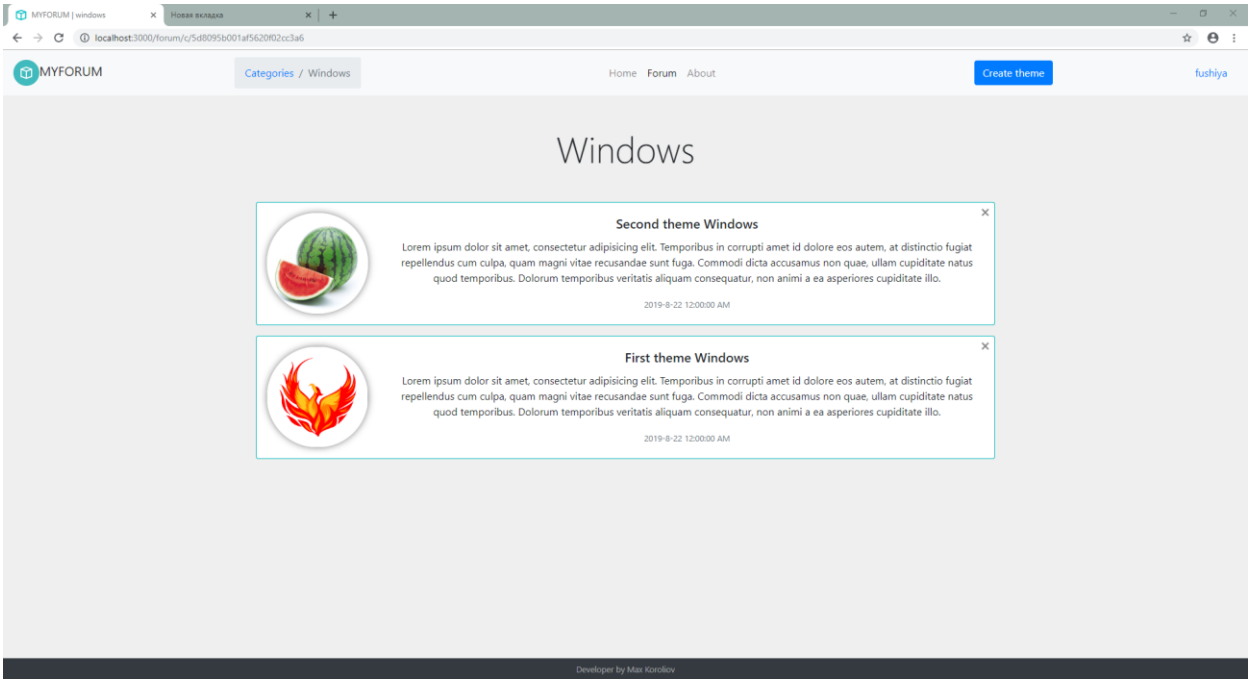

Рисунок 3.2.3 – Сторінка зі списком тем

На сторінці тем, у навігаційній панелі з'явився новий елемент – кнопка «Створити тему». Створювати тему можуть тільки авторизовані користувачі, в іншому випадку здісниться перенаправлення на головну сторінку.

Список тем сортується від найновішого, до найранішого створеного. Переконатись у цьому можна, глянувши під опис теми – там відображається дата створення теми.

Обравши тему та на неї натиснувши, сторінка відобразить список усіх повідомлень даної теми, сортуючи від найпершого повідомлення до найновішого.

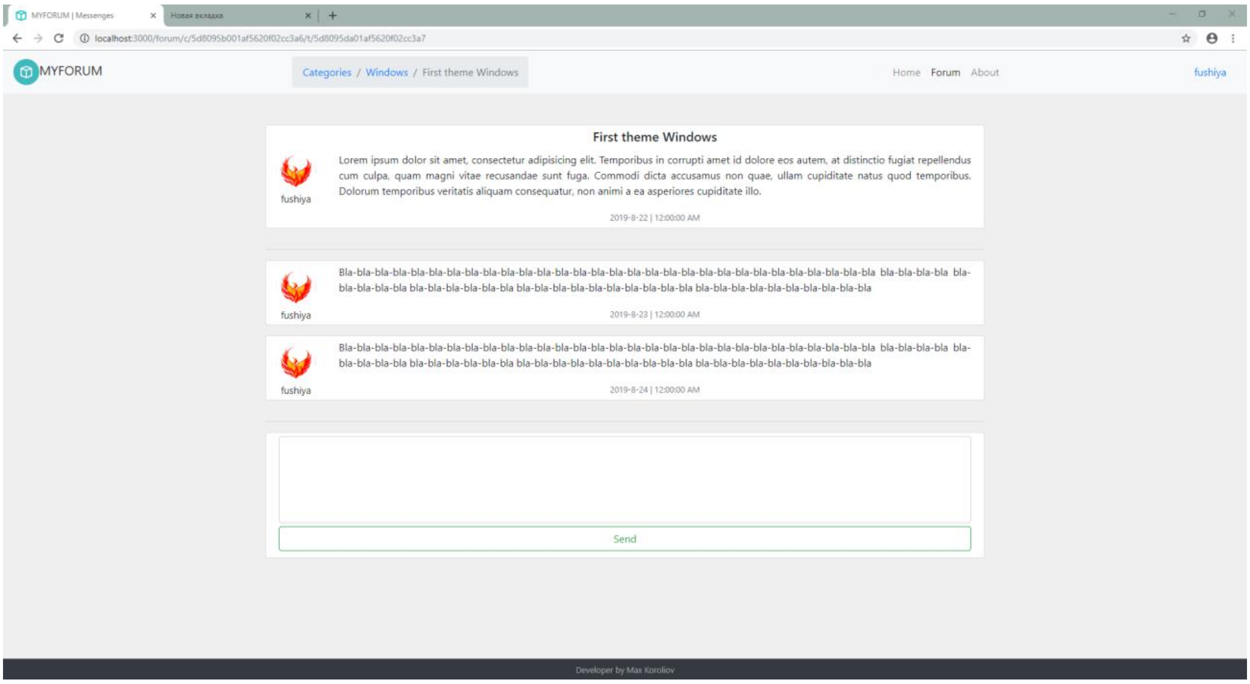

Рисунок 3.2.4 – Сторінка повідомлень теми.

Якщо користувач не авторизований, то він не зможе відправити повідомлення.

Якщо натиснути на логін, який знаходиться на навігаційній панелі, то відкриється особистий кабінет користувача, у якому можна редагувати дані користувача.

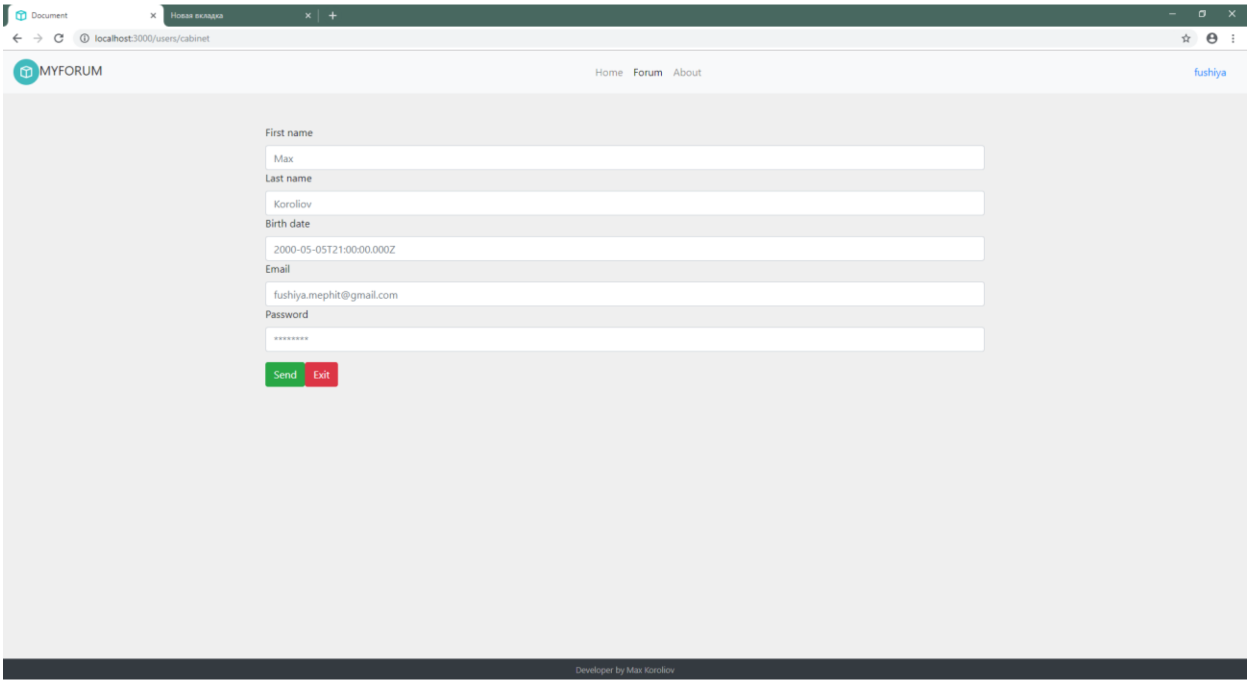

Рисунок 3.2.5 – Особистий кабінет користувача

Для того, щоб створити категорію, необхідно перейти на адміністративну панель. Щоб потрапити на адміністративну панель, необхідно перейти на сторінку /admin та бути авторизованим користувачем з правами адміністратора. В іншому випадку здійсниться перенаправлення на головну сторінку. У базі даних на даний момент зареєстровано всього три користувача - права адміністратора має тільки користувач з логіном “fushiya”.

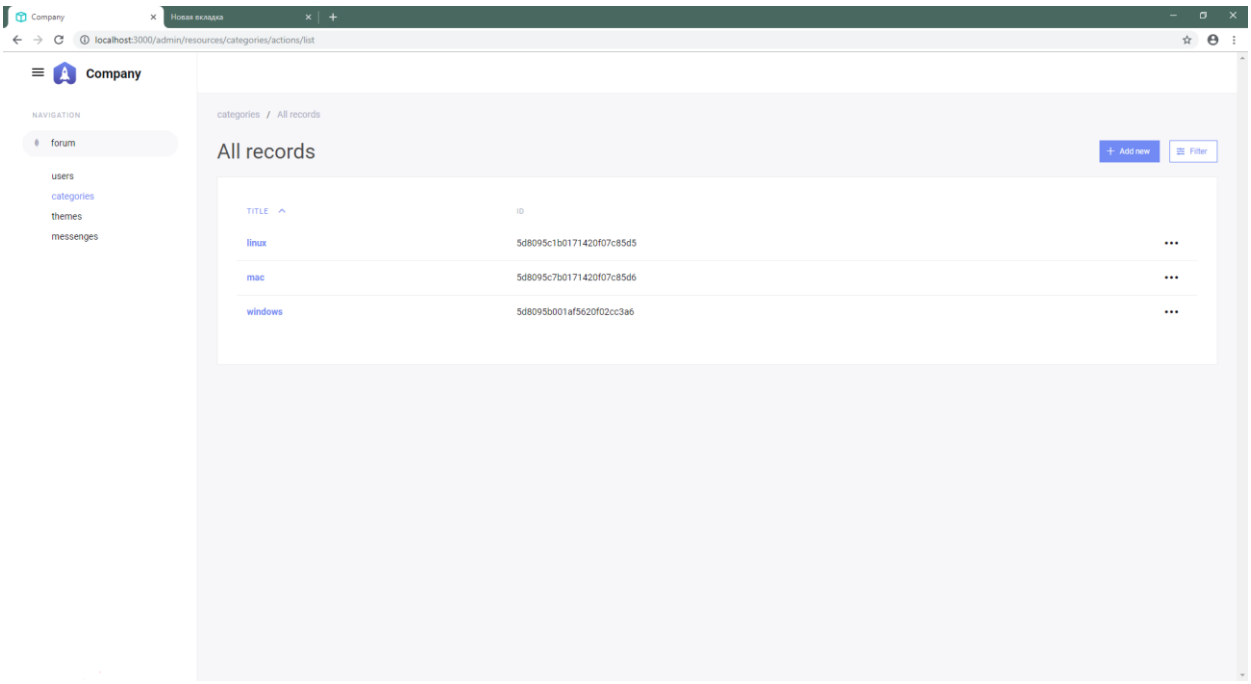

Рисунок 3.2.6 – Панель адміністратора

На панелі адміністратора, також є можливість редагувати та контролювати будь-яку інформацію на даному ресурсі.

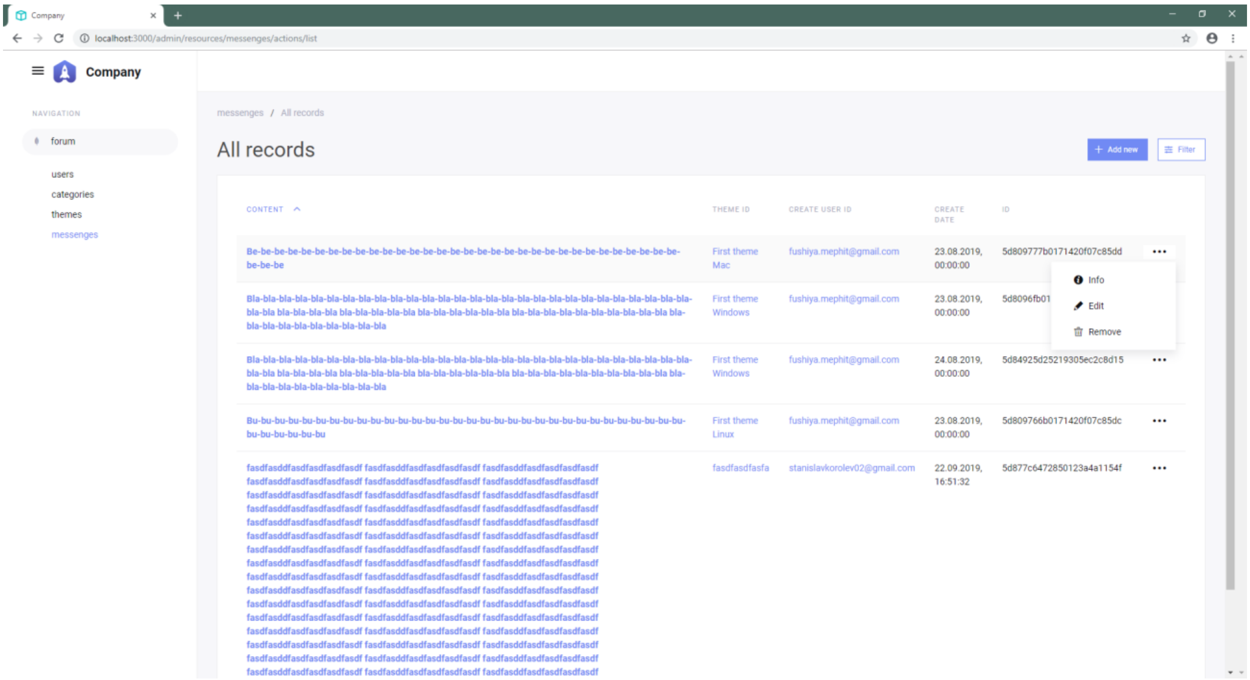

Рисунок 3.2.7 – Редагування повідомлень учасників форума

ВИСНОВКИ

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

Розроблена система забезпечує виконання функцій веб-форума та здатність контролювати систему за допомогою адміністративної панелі.

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 В. /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 Е. /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_userI
d']).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.conten
t_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'}, {p
ath: '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 Н. 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 І. 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 О. 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 Р. 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);
```