МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)**

Институт информационных технологий математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»

ОТЧЕТ

По проектной работе

На тему:
Заметки с синхронизацией через сервер на Flask

Выполнил: студент группы 381806-2

_____ Напылов Е.И.
Подпись

Проверил:

_____ Карчков Д.А.
Подпись

Нижний Новгород
2021

Содержание

1. Введение

Sailfish OS – мобильная операционная система с открытым исходным кодом. Разработка приложений под данную ОС ведется с помощью Sailfish SDK, который обладает широким спектром возможностей. Интерфейс построен на фреймворке QT, разработка ведется на языке QML, близкому к JavaScript. В качестве инструмента используется IDE QT Creator.

2. Постановка задачи

Требуется разработать приложение, позволяющее создавать, изменять и удалять заметки. Мобильная (клиентская) часть приложения должна быть разработана на языке qml под платформу Sailfish OS. Для синхронизации необходимо разработать серверное приложение. В качестве языка был выбран Python, с использованием микрофреймворка Flask. Кроме мобильного клиента необходимо разработать версию для браузера, которая, например, может использоваться на ПК.

3. Разработка серверной части

3.1 Работа с базой данных

Для хранения данных – заметок и информации о пользователях была выбрана база данных sqlite3. Sqlite – база данных, подходящая для небольших или тестовых проектов, т.к. вся информация хранится в единственном файле, отсутствует необходимость использования отдельного сервера для базы данных.

Использование SQL запросов в чистом виде – неудобное решение. Поэтому для работы с базой данных я выбрал ORM SQLAlchemy. Эта библиотека инкапсулирует работу с SQL для разных баз данных. Программист описывает схему базу данных в виде классов и работает с базой на уровне языка Python.

Схема таблицы пользователей:

```
class Users(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(50), unique=True)
    username = db.Column(db.String(0), unique=True)
    password = db.Column(db.String(500), nullable=False)
    date_reg = db.Column(db.DateTime, default=datetime.utcnow)
    date_upd = db.Column(db.DateTime, default=datetime.utcnow)

    def __repr__(self):
        return f"<User: id: {self.id}, email: {self.email}, username: {self.username}, date: {self.date_reg}"
```

UserMixin – класс для работы с учетными записями из плагина Flask Login.

Схема таблицы с заметками:

```
class Notes(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    title = db.Column(db.Text, nullable=False)
    detail_text = db.Column(db.Text)
    date_creat = db.Column(db.DateTime, default=datetime.utcnow)
    date_upd = db.Column(db.DateTime, default=datetime.utcnow)

    def __repr__(self):
        return f"<Note: id: {self.id}, user_id: {self.user_id}, title: {self.title}, date: {self.date_creat}"
```

3.2 API для работы с учетными записями пользователей

Регистрация.

Регистрация реализована в браузерной версии, т.к. в мобильном приложении пользователю будет неудобно заполнять большое количество полей и подтверждать адрес электронной почты.

Для регистрации используются следующие данные: username, password, email.

В базе данных пароль хранится в виде хэша, таким образом, никто кроме пользователя его не знает.

```
hash_pass = generate_password_hash(password)
user = Users(email=email, username=username, password=hash_pass)
db.session.add(user)
db.session.flush()
db.session.commit()
flash('User was created. Please sign in.', category='ok')
return redirect(url_for('login'))
```

Авторизация.

Авторизация производится по паре username-password. Вычисляется хэш введенного пароля и сравнивается с хэшом из базы данных.

```
user = Users.query.filter_by(username=username).first()
if user and check_password_hash(user.password, password):
    login_user(user, remember=remember_me)
    return redirect(url_for('notes'))
```

Для аутентификации пользователя в мобильном API используется userid, который приходит как ответ из эндпоинта api/login.

```
@app.route('/api/login', methods=['POST'])
def api_login():
    data = request.get_json()
    _login = data['login']
    _password = data['password']

    user = Users.query.filter_by(username=_login).first()

    userid = -1
    username = ""
    success = False

    if user and check_password_hash(user.password, _password):
        userid = user.id
        username = user.username
        success = True
    resp = {
        'success': success,
        'userid': userid,
        'username': username
    }
    return jsonify(resp)
```

3.3 API для работы с заметками

Получение заметок.

На сервере реализованы 2 способа получения заметок:

1. Все заметки пользователя

```python
@app.route('/api/get_all_notes', methods=['POST'])
def api_get_all_notes():
    data = request.get_json()
    _userid = data['userid']
    all_notes = get_all_notes(_userid)
    resp = {
        'notes': all_notes,
    }
    return jsonify(resp)
```

2. Конкретная заметка по ее id

```python
@app.route('/api/get_one_note_by_id', methods=['POST'])
def api_get_one_note_by_id():
    data = request.get_json()
    _userid = data['userid']
    _note_id = data['note_id']

    _note = Notes.query.filter_by(user_id=_userid, id=_note_id).first()

    resp = {
        "title": _note.title,
        "text": _note.detail_text,
        "date": _note.date_creat,
        "id": _note.id,
    }
    return jsonify(resp)
```

Создание новой заметки.

Для создания новой заметки необходимо в json передать title, text и userid.

```
@app.route('/api/add_note', methods=['POST'])
def api_add_note():
    data = request.get_json()
    _userid = data['userid']
    _title = data['title']
    _text = data['text']

    success = False
    if _title != '':
        note = Notes(user_id=_userid, title=_title, detail_text=_text)
        db.session.add(note)
        db.session.flush()
        db.session.commit()
        success = True

    resp = {
        'success': success,
    }

    return jsonify(resp)
```

Удаление заметки

Удаление заметки производится по ее id, userid необходим для контроля доступа.

```
@app.route('/api/del_note', methods=['POST'])
def api_del_note():
    data = request.get_json()
    _userid = data['userid']
    _note_id = data['note_id']

    Notes.query.filter_by(user_id=_userid, id=_note_id).delete()
    db.session.flush()
    db.session.commit()

    success = True

    resp = {
        'success': success,
    }

    return jsonify(resp)
```

Редактирование заметки.

Редактирование происходит по id заметки. В качестве параметров передаются новое название и содержание.

```
@app.route('/api/edit_note', methods=['POST'])
def api_edit_note():
    data = request.get_json()
    _userid = data['userid']
    _title = data['title']
    _text = data['text']
    _note_id = data['note_id']

    success = False
    if _title != '':
        Notes.query.filter_by(user_id=_userid, id=_note_id).update(
            {'title': _title, 'detail_text': _text, 'date_upd': datetime.utcnow()})
        db.session.flush()
        db.session.commit()
        success = True

    resp = {
        'success': success,
    }
```

4. Версия для браузера

Поскольку разработка веб приложения не относится к курсу напрямую, приведу только краткое описание.

Чтобы не тратить время на разработку клиентского JS кода я использовал подход с шаблонами с рендером на стороне сервера.

В качестве шаблонизатора был использован стандартный для Flask Jinja2.

Пример работы с шаблонизатором:

```
@app.route('/notes/<note_id>', methods=['GET'])
@login_required
def note_detail(note_id):
    note = Notes.query.filter_by(user_id=current_user.id, id=note_id).first()
    return render_template('details.html', note=note)
```

Для корректного отображения даты в часовом поясе клиента была использована небольшая библиотека Flask Moment. Библиотека вставляет в HTML страницу небольшой код на JS, преобразующий стандартное время сервера под часовой пояс клиента.

```
</head>
{{ moment.include_moment() }}
<body>
```

Для быстрого создания приятного интерфейса был использован CSS «фреймворк» Bootstrap.

5. Мобильное приложение

5.1 Страница со списком заметок.

Получение заметок с сервера происходит по ранее описанному API. Заметки складываются в модель ListModel.

```
var json_resp = JSON.parse(xmlHttp.responseText)
for (var i = 0; i < json_resp['notes'].length; i++) {
    var _title = json_resp['notes'][i]['title']
    var _text = json_resp['notes'][i]['text']
    var _date = json_resp['notes'][i]['date']
    var _id = json_resp['notes'][i]['id']
    notes_model.append({date: _date, title: _title, text: _text, id: _id})
}
```

Для отображения модели используется компонент SilicaListView.

```
delegate: BackgroundItem {
    id: delegate

    Label {
        x: Theme.horizontalPageMargin
        text: title + "\n" + date
        anchors.verticalCenter: parent.verticalCenter
    }

    onClicked: {
        console.log("Clicked note id: " + id)
        current_note_id.value = id
        console.log("Current note id cfg: " + current_note_id.value)
        pageStack.animatorPush(Qt.resolvedUrl("NoteDetails.qml"))
    }
}
VerticalScrollDecorator {}
```

В качестве делегата используется BackGroundItem, добавляющий возможность нажатия на элемент. В метку выводится название и дата создания заметки.

В событии onClicked в ConfigurationValue запоминается id нажатой выбранной заметки и происходит переход на страницу с детальным описанием и возможностью редактирования.

5.2 Страница с содержанием заметки.

На странице расположены 3 текстовых поля – дата, заголовок и содержание.

Данные конкретной заметки подгружаются по api/get_one_note_by_id.

Поля заголовок и содержания можно редактировать.

```
Label {
    id: note_date
    text: "note_date"
    anchors.horizontalCenter: parent.horizontalCenter
}
```

```
TextArea {
    id: note_title
    text: "note title"
    background: Rectangle {
        color: "black"
        opacity: 0.2
        width: parent.width
        height: parent.height
    }
}
```

```
TextArea {
    id: note_text
    text: "note_text"
    wrapMode: Text.WrapAnywhere
    background: Rectangle {
        color: "black"
        opacity: 0.2
        width: parent.width
        height: parent.height
    }
}
```

На странице есть 2 кнопки – сохранить изменения и удалить. Действия происходят с использованием соответствующих url API.

```
function get_note_by_id(_userid, _note_id) {
    const xmlHttp = new XMLHttpRequest()
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
            var json_resp = JSON.parse(xmlHttp.responseText)
            var title = json_resp['title']
            var text = json_resp['text']
            var date = json_resp['date']
            var id = json_resp['id']
            note_date.text = date
            note_title.text = title
            note_text.text = text
        }
    }
    xmlHttp.open("POST", host_cfg_value.value+"/api/get_one_note_by_id", true)
    xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
    var response_json = JSON.stringify({"userid": _userid, "note_id": _note_id});
    xmlHttp.send(response_json)
}
```

```
function edit_note(_userid, _note_id, _title, _text) {
    const xmlHttp = new XMLHttpRequest()
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
            var json_resp = JSON.parse(xmlHttp.responseText)
            console.log(json_resp['success'])
        }
    }
    xmlHttp.open("POST", host_cfg_value.value+"/api/edit_note", true)
    xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
    var response_json = JSON.stringify({
                                           "userid": _userid,
                                           "title": _title,
                                           "text": _text,
                                           "note_id": _note_id
                                       });
    xmlHttp.send(response_json)
}
```

```
function delete_note(_userid, _note_id) {
    const xmlHttp = new XMLHttpRequest()
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
            var json_resp = JSON.parse(xmlHttp.responseText)
            console.log(json_resp['success'])
        }
    }
    xmlHttp.open("POST", host_cfg_value.value+"/api/del_note", true)
    xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
    var response_json = JSON.stringify({
                                           "userid": _userid,
                                           "note_id": _note_id
                                       });
    xmlHttp.send(response_json)
}
```

5.3 Страница для создания заметки.

На странице расположены 2 текстовых поля – заголовок и содержание.

Сохранение заметки происходит с использованием url api/add_note.

```
TextArea {
    id: note_title
    text: "Title"
    background: Rectangle {
        color: "black"
        opacity: 0.2
        width: parent.width
        height: parent.height
    }
}
```

```
TextArea {
    id: note_text
    text: "Text"
    wrapMode: Text.WrapAnywhere
    background: Rectangle {
        color: "black"
        opacity: 0.2
        width: parent.width
        height: parent.height
    }
}
```

```
function add_new_note(_userid, _title, _text) {
    const xmlHttp = new XMLHttpRequest()
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
            var json_resp = JSON.parse(xmlHttp.responseText)
            console.log(json_resp['success'])
        }
    }
    xmlHttp.open("POST", host_cfg_value.value+"/api/add_note", true)
    xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
    var response_json = JSON.stringify({
                                           "userid": _userid,
                                           "title": _title,
                                           "text": _text
                                       });
    xmlHttp.send(response_json)
}
```

5.4 Страница настроек.

У страницы настроек 2 функции.

Авторизация.

Авторизация происходит по паре username-password. При успешной авторизации сервер возвращает в ответе userid. Username и userid сохраняются в памяти приложения с использованием ConfigurationValue.

```
function login(_login, _password){
    const xmlHttp = new XMLHttpRequest()
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
            var json_resp = JSON.parse(xmlHttp.responseText)
            var tmp = "UserId\n"
            userid_cfg_value.value = json_resp['userid']
            username_cfg_value.value = json_resp['username']
            status_label.text = "Login status: " + json_resp['success'] + ", as " + json_resp['username']
        }
    }
    xmlHttp.open("POST", host_cfg_value.value+"/api/login", true)
    xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
    var response_json = JSON.stringify({"login": _login,
                                        "password": _password,
                                       });
    xmlHttp.send(response_json)
}
```

```
ConfigurationValue {
    id: userid_cfg_value
    key: "/apps/NotesMobileClient/userid"
    defaultValue: -1
}

ConfigurationValue {
    id: username_cfg_value
    key: "/apps/NotesMobileClient/username"
    defaultValue: ""
}
```

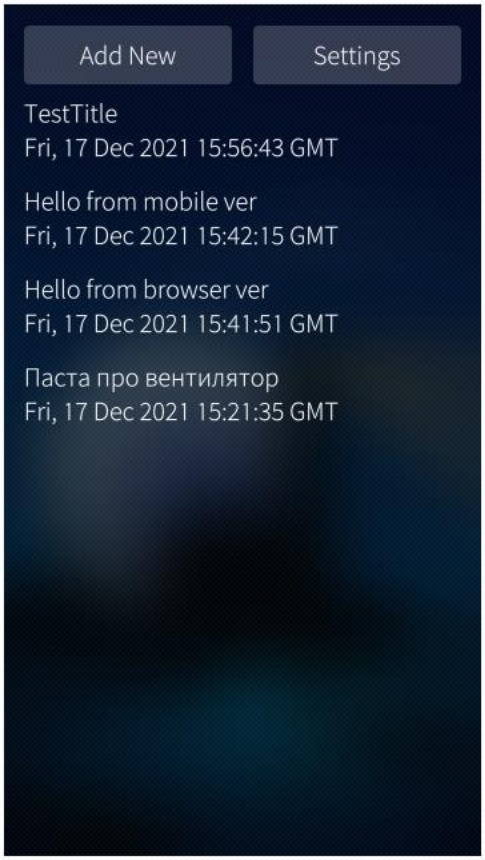
Изменение хоста.

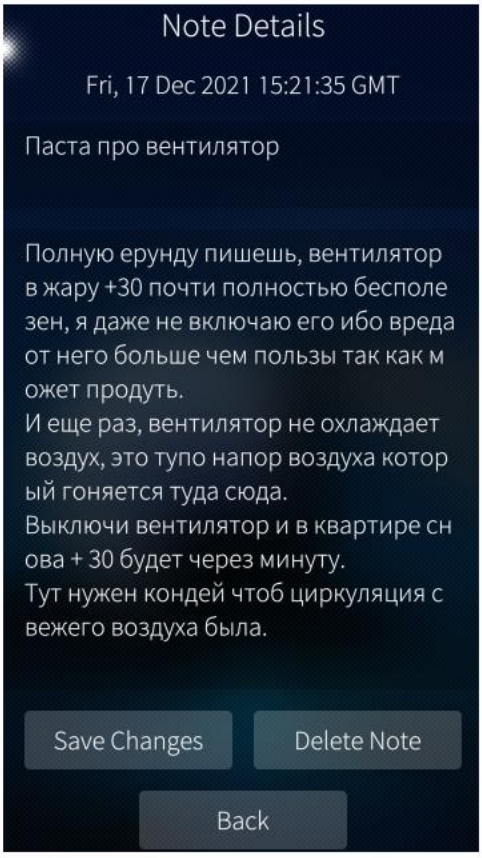
Часто случается, что сервер переходит на новый IP адрес, а в процессе разработки это происходит постоянно. Для решения этой проблемы предусмотрен параметр с именем хоста. Значение сохраняется в ConfigurationValue.

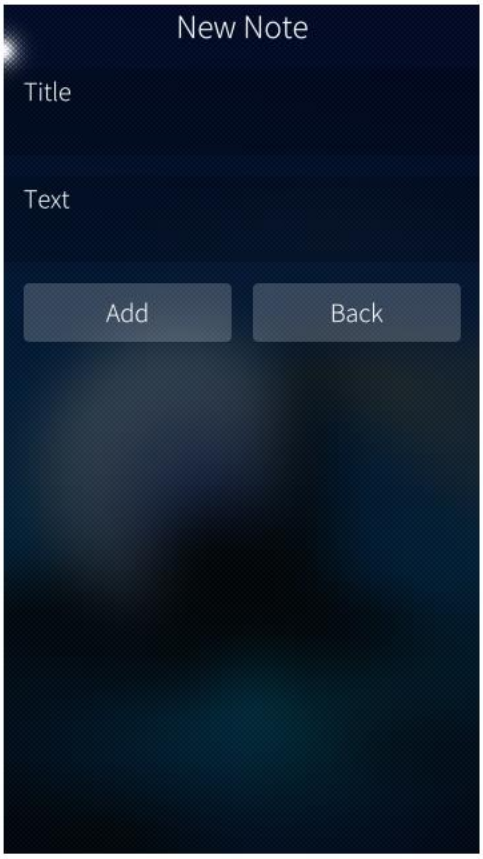
```
ConfigurationValue {
    id: host_cfg_value
    key: "/apps/NotesMobileClient/host"
    defaultValue: ""
}
```

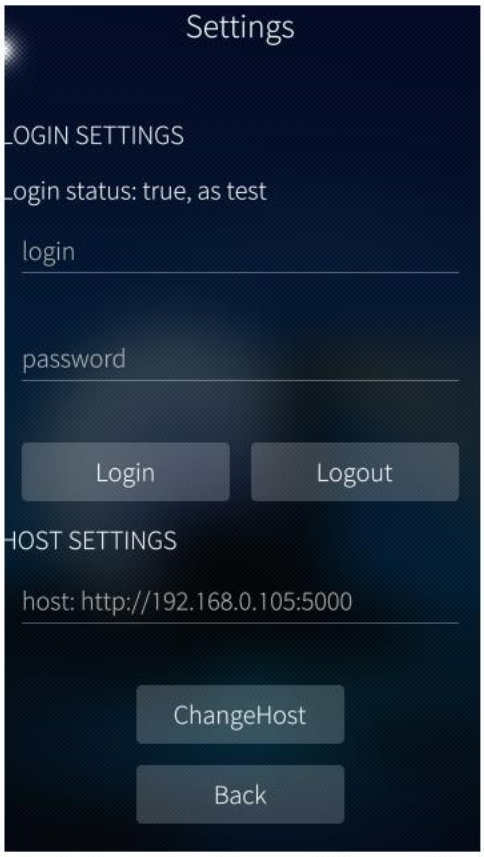
6. Демонстрация приложения

6.1 Мобильное приложение.

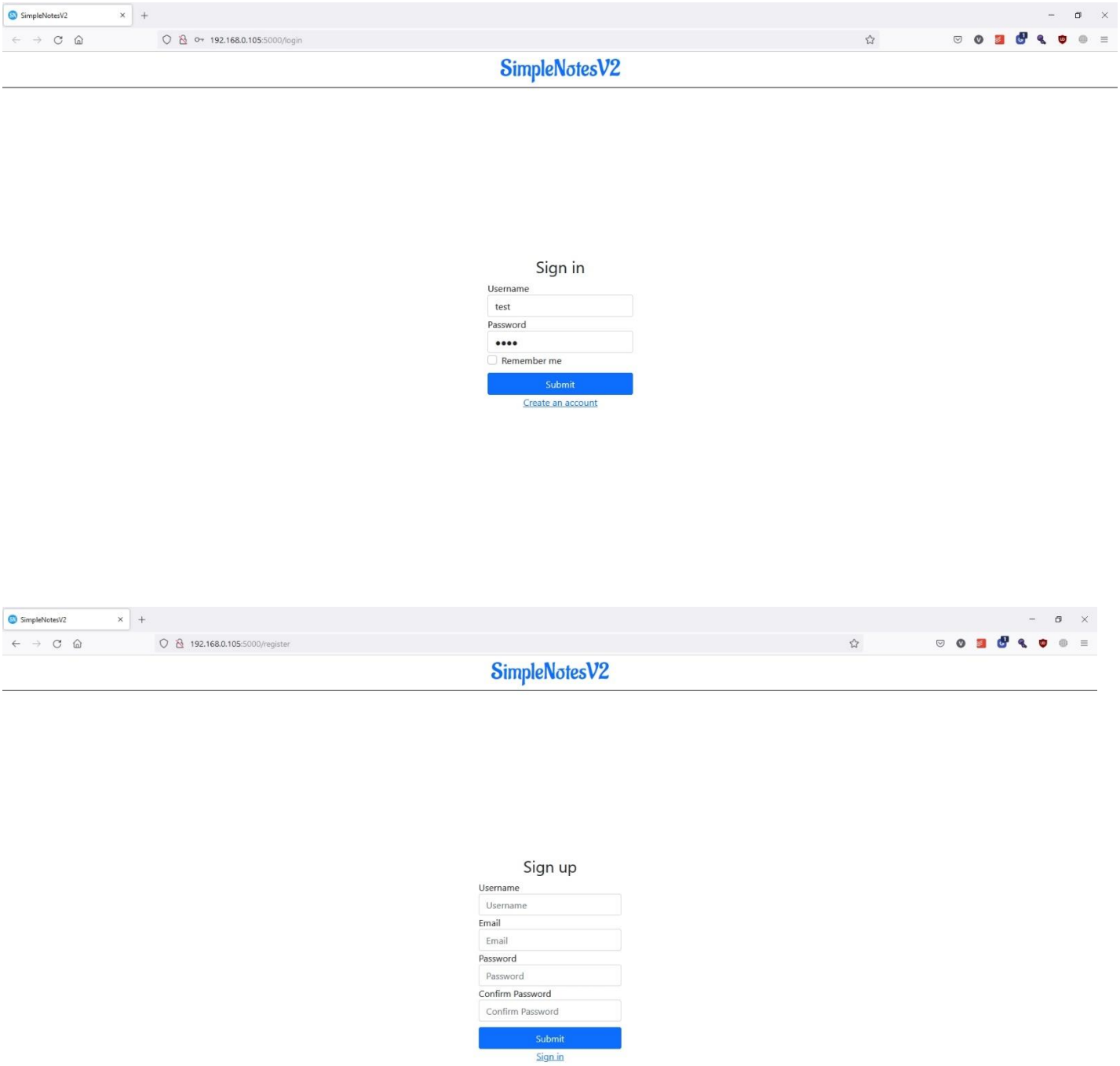








6.2 Версия для браузера




SimpleNotesV2
192.168.0.105:5000
SimpleNotesV2
test (Log out) (minigame)
Notes
New
TestTitle
December 17, 2021 6:56 PM
Hello from mobile ver
December 17, 2021 6:42 PM
Hello from browser ver
December 17, 2021 6:41 PM
Паста про вентилятор
December 17, 2021 6:21 PM


SimpleNotesV2
192.168.0.105:5000/notes/21
SimpleNotesV2
test (Log out) (minigame)
Note details
Паста про вентилятор
December 17, 2021 6:21 PM
Полную ерунду пишешь, вентилятор в жару +30 почти полностью бесполезен, я даже не включаю его ибо вреда от него больше чем пользы так как может продуть.
И еще раз, вентилятор не охлаждает воздух, это тупо напор воздуха который гоняется туда сюда.
Выключи вентилятор и в квартире снова + 30 будет через минуту.
Тут нужен кондей чтоб циркуляция свежего воздуха была.
Edit
Delete

7. Заключение

В ходе работы было разработано приложение с использованием клиент-серверной архитектуры.

Сервер написан на языке Python с использованием фреймворка Flask. На стороне сервера была разработана схема хранения заметок и учетных записей пользователя. Также было создано Json API, которое используется мобильным приложением.

После создания сервера я разработал мобильный клиент под платформу Sailfish OS с использованием языка программирования qml. В мобильном приложении реализованы функции: авторизация, просмотр списка заметок, создание, редактирование и удаление. Все функции синхронизируются с сервером с помощью Json API.

А также был разработан фронтенд для браузерной версии приложения.

8. Исходный код

8.1 Сервер

// models.py

```
from datetime import datetime

from flask_login import UserMixin

from simple_notes import db


class Users(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(50), unique=True)
    username = db.Column(db.String(0), unique=True)
    password = db.Column(db.String(500), nullable=False)
    date_reg = db.Column(db.DateTime, default=datetime.utcnow)
    date_upd = db.Column(db.DateTime, default=datetime.utcnow)

    def __repr__(self):
        return f"<User: id: {self.id}, email: {self.email}, username: {self.username},
date: {self.date_reg}"


class Notes(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    title = db.Column(db.Text, nullable=False)
    detail_text = db.Column(db.Text)
    date_creat = db.Column(db.DateTime, default=datetime.utcnow)
    date_upd = db.Column(db.DateTime, default=datetime.utcnow)

    def __repr__(self):
        return f"<Note: id: {self.id}, user_id: {self.user_id}, title: {self.title}, date:
{self.date_creat}"
```

// routes.py

```
from datetime import datetime

from flask import request, redirect, url_for, render_template, flash, jsonify
from flask_login import login_user, login_required, logout_user, current_user
from werkzeug.security import generate_password_hash, check_password_hash

from simple_notes import app, db, login_manager
from simple_notes.models import Users, Notes


@login_manager.unauthorized_handler
def unauthorized_callback():
    # if unauthorized then redirect to login
```

```
    return redirect(url_for('login'))


@login_manager.user_loader
def load_user(user_id):
    # load user by id from database
    return Users.query.filter_by(id=user_id).first()


@app.route('/register', methods=("POST", "GET"))
def register():
    if current_user.is_authenticated:
        print(current_user)
        return redirect(url_for('notes'))
    if request.method == 'POST':
        email = request.form['email']
        username = request.form['username']
        password = request.form['password']
        password_2 = request.form['password_2']

        if password == password_2 and password != '' and email != '' and username != '':
            try:
                hash_pass = generate_password_hash(password)
                user = Users(email=email, username=username, password=hash_pass)
                db.session.add(user)
                db.session.flush()
                db.session.commit()
                flash('User was created. Please sign in.', category='ok')
                return redirect(url_for('login'))
            except Exception as e:
                db.session.rollback()
                print(e)
                flash('Error in the registration form', category='err')
    return render_template('register.html')


@app.route('/login', methods=['POST', 'GET'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('notes'))
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        remember_me = True if request.form.get("remember-me") else False

        user = Users.query.filter_by(username=username).first()
        if user and check_password_hash(user.password, password):
            login_user(user, remember=remember_me)
            return redirect(url_for('notes'))
        else:
            print('fail login')
            flash('Password and login do not match', category='err')
            pass
    return render_template('login.html')
```

```python
@app.route('/logout', methods=['POST', 'GET'])
@login_required
def logout():
    logout_user()
    return redirect(url_for('notes'))


@app.route('/', methods=['GET'])
@login_required
def notes():
    curr_user_notes =
Notes.query.filter_by(user_id=current_user.id).order_by(Notes.date_creat.desc()).all()
    return render_template('notes.html', notes=curr_user_notes)


@app.route('/add_note', methods=['POST'])
@login_required
def add_note():
    title = request.form['title']
    details = request.form['details']
    if title != '':
        note = Notes(user_id=current_user.id, title=title, detail_text=details)
        db.session.add(note)
        db.session.flush()
        db.session.commit()
    return redirect(url_for('notes'))


@app.route('/notes/<note_id>', methods=['GET'])
@login_required
def note_detail(note_id):
    note = Notes.query.filter_by(user_id=current_user.id, id=note_id).first()
    return render_template('details.html', note=note)


@app.route('/delete_note/<note_id>', methods=['GET'])
@login_required
def delete_note(note_id):
    Notes.query.filter_by(user_id=current_user.id, id=note_id).delete()
    db.session.flush()
    db.session.commit()
    return redirect(url_for('notes'))


@app.route('/edit_note/<note_id>', methods=['POST'])
@login_required
def edit_note(note_id):
    title = request.form['title']
    details = request.form['details']
    if title != '':
        Notes.query.filter_by(user_id=current_user.id, id=note_id).update(
            {'title': title, 'detail_text': details, 'date_upd': datetime.utcnow()})
```

```
        db.session.flush()
        db.session.commit()
    return redirect(url_for('note_detail', note_id=note_id))


@app.route('/minigame', methods=['GET'])
@login_required
def mini_game():
    return render_template('minigame.html')


# ######################################### API
#########################################

def get_all_notes(userid):
    user_notes =
Notes.query.filter_by(user_id=userid).order_by(Notes.date_creat.desc()).all()
    all_notes = []
    for _note in user_notes:
        all_notes.append({
            "title": _note.title,
            "text": _note.detail_text,
            "date": _note.date_creat,
            "id": _note.id,
        })
    return all_notes


@app.route('/api/login', methods=['POST'])
def api_login():
    data = request.get_json()
    _login = data['login']
    _password = data['password']

    user = Users.query.filter_by(username=_login).first()

    userid = -1
    username = ""
    success = False

    if user and check_password_hash(user.password, _password):
        userid = user.id
        username = user.username
        success = True
    resp = {
        'success': success,
        'userid': userid,
        'username': username
    }
    return jsonify(resp)


@app.route('/api/get_all_notes', methods=['POST'])
def api_get_all_notes():
```

```
    data = request.get_json()
    _userid = data['userid']
    all_notes = get_all_notes(_userid)
    resp = {
        'notes': all_notes,
    }
    return jsonify(resp)


@app.route('/api/add_note', methods=['POST'])
def api_add_note():
    data = request.get_json()
    _userid = data['userid']
    _title = data['title']
    _text = data['text']

    success = False
    if _title != '':
        note = Notes(user_id=_userid, title=_title, detail_text=_text)
        db.session.add(note)
        db.session.flush()
        db.session.commit()
        success = True

    resp = {
        'success': success,
    }

    return jsonify(resp)


@app.route('/api/del_note', methods=['POST'])
def api_del_note():
    data = request.get_json()
    _userid = data['userid']
    _note_id = data['note_id']

    Notes.query.filter_by(user_id=_userid, id=_note_id).delete()
    db.session.flush()
    db.session.commit()

    success = True

    resp = {
        'success': success,
    }

    return jsonify(resp)


@app.route('/api/edit_note', methods=['POST'])
def api_edit_note():
    data = request.get_json()
    _userid = data['userid']
```

```
    _title = data['title']
    _text = data['text']
    _note_id = data['note_id']

    success = False
    if _title != '':
        Notes.query.filter_by(user_id=_userid, id=_note_id).update(
            {'title': _title, 'detail_text': _text, 'date_upd': datetime.utcnow()})
        db.session.flush()
        db.session.commit()
        success = True

    resp = {
        'success': success,
    }

    return jsonify(resp)


@app.route('/api/get_one_note_by_id', methods=['POST'])
def api_get_one_note_by_id():
    data = request.get_json()
    _userid = data['userid']
    _note_id = data['note_id']

    _note = Notes.query.filter_by(user_id=_userid, id=_note_id).first()

    resp = {
        "title": _note.title,
        "text": _note.detail_text,
        "date": _note.date_creat,
        "id": _note.id,
    }
    return jsonify(resp)
```

8.2 Мобильное приложение

// FirstPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import Nemo.Configuration 1.0

Page {

    ConfigurationValue {
        id: userid_cfg_value
        key: "/apps/NotesMobileClient/userid"
        defaultValue: -1
    }
    ConfigurationValue {
        id: host_cfg_value
```

```
    key: "/apps/NotesMobileClient/host"
    defaultValue: ""
}

ConfigurationValue {
    id: current_note_id
    key: "/apps/NotesMobileClient/current_note_id"
    defaultValue: -1
}


function get_all_notes(_userid) {
    const xmlHttp = new XMLHttpRequest()
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
            var json_resp = JSON.parse(xmlHttp.responseText)
            for (var i = 0; i < json_resp['notes'].length; i++) {
                var _title = json_resp['notes'][i]['title']
                var _text = json_resp['notes'][i]['text']
                var _date = json_resp['notes'][i]['date']
                var _id = json_resp['notes'][i]['id']
                notes_model.append({date: _date, title: _title, text: _text, id: _id})
            }
        }
    }
    xmlHttp.open("POST", host_cfg_value.value+"/api/get_all_notes", true)
    xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
    var response_json = JSON.stringify({"userid": _userid});
    xmlHttp.send(response_json)
}

id: page

allowedOrientations: Orientation.All

ListModel {
    id: notes_model
}

SilicaListView {
    id: listView
    model: notes_model
    anchors.fill: parent
    spacing: Theme.paddingLarge

    header: PageHeader {
        title: qsTr("")
    }

    Button {
        id: new_note_bt
        x: Theme.paddingLarge
        y: Theme.paddingLarge
        text: "Add New"
```

```
            onClicked: {
                pageStack.animatorPush(Qt.resolvedUrl("NewNote.qml"))
            }
        }

        Button {
            id: settings_bt
            y: Theme.paddingLarge
            x: new_note_bt.x + new_note_bt.width + Theme.paddingLarge
            text: "Settings"
            onClicked: {
                pageStack.animatorPush(Qt.resolvedUrl("Settings.qml"))
            }
        }

        delegate: BackgroundItem {
            id: delegate

            Label {
                x: Theme.horizontalPageMargin
                text: title + "\n" + date
                anchors.verticalCenter: parent.verticalCenter
            }

            onClicked: {
                console.log("Clicked note id: " + id)
                current_note_id.value = id
                console.log("Current note id cfg: " + current_note_id.value)
                pageStack.animatorPush(Qt.resolvedUrl("NoteDetails.qml"))
            }
        }
        VerticalScrollDecorator {}
    }

    Timer {
        id: timer
        interval: 60000
        repeat: true
        running:true
        onTriggered:  {
            notes_model.clear();
            get_all_notes(userid_cfg_value.value)
        }
    }

    Component.onCompleted: {get_all_notes(userid_cfg_value.value)}
}
```

// NoteDetails.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import Nemo.Configuration 1.0
```

```
Page {

    ConfigurationValue {
        id: userid_cfg_value
        key: "/apps/NotesMobileClient/userid"
        defaultValue: -1
    }
    ConfigurationValue {
        id: host_cfg_value
        key: "/apps/NotesMobileClient/host"
        defaultValue: ""
    }

    ConfigurationValue {
        id: current_note_id
        key: "/apps/NotesMobileClient/current_note_id"
        defaultValue: -1
    }

    function get_note_by_id(_userid, _note_id) {
        const xmlHttp = new XMLHttpRequest()
        xmlHttp.onreadystatechange = function () {
            if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
                var json_resp = JSON.parse(xmlHttp.responseText)
                var title = json_resp['title']
                var text = json_resp['text']
                var date = json_resp['date']
                var id = json_resp['id']
                note_date.text = date
                note_title.text = title
                note_text.text = text
            }
        }
        xmlHttp.open("POST", host_cfg_value.value+"/api/get_one_note_by_id", true)
        xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
        var response_json = JSON.stringify({"userid": _userid, "note_id": _note_id});
        xmlHttp.send(response_json)
    }

    function edit_note(_userid, _note_id, _title, _text) {
        const xmlHttp = new XMLHttpRequest()
        xmlHttp.onreadystatechange = function () {
            if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
                var json_resp = JSON.parse(xmlHttp.responseText)
                console.log(json_resp['success'])
            }
        }
        xmlHttp.open("POST", host_cfg_value.value+"/api/edit_note", true)
        xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
        var response_json = JSON.stringify({
                                               "userid": _userid,
                                               "title": _title,
                                               "text": _text,
                                               "note_id": _note_id
```

```
                                    });
    xmlHttp.send(response_json)
}

function delete_note(_userid, _note_id) {
    const xmlHttp = new XMLHttpRequest()
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
            var json_resp = JSON.parse(xmlHttp.responseText)
            console.log(json_resp['success'])
        }
    }
    xmlHttp.open("POST", host_cfg_value.value+"/api/del_note", true)
    xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
    var response_json = JSON.stringify({
                                        "userid": _userid,
                                        "note_id": _note_id
                                    });
    xmlHttp.send(response_json)
}



id: page
allowedOrientations: Orientation.All
SilicaFlickable {
    anchors.fill: parent

    contentHeight: column.height



    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge

        Label {
            id: header
            text: "Note Details"
            font.pixelSize: 50
            anchors.horizontalCenter: parent.horizontalCenter
        }


            Label {
                id: note_date
                text: "note_date"
                anchors.horizontalCenter: parent.horizontalCenter
            }


        TextArea {
            id: note_title
```

```
            text: "note title"
            background: Rectangle {
                color: "black"
                opacity: 0.2
                width: parent.width
                height: parent.height
            }
        }

        TextArea {
            id: note_text
            text: "note_text"
            wrapMode: Text.WrapAnywhere
            background: Rectangle {
                color: "black"
                opacity: 0.2
                width: parent.width
                height: parent.height
            }

        }


        Row {
            x: Theme.paddingLarge
            spacing: Theme.paddingLarge
            id: row_controls
            Button {
                id: bt_save_changes
                text: "Save Changes"
                onClicked: {
                    edit_note(userid_cfg_value.value, current_note_id.value,
note_title.text, note_text.text)
                }
            }

            Button {
                id: delete_note_bt
                text: "Delete Note"
                onClicked: {
                    delete_note(userid_cfg_value.value, current_note_id.value)
                    pageStack.animatorPush(Qt.resolvedUrl("FirstPage.qml"))
                }
            }
        }

        Button {
            id: bt_to_all_notes_page
            anchors.horizontalCenter: row_controls.horizontalCenter
            text: "Back"
             onClicked: pageStack.animatorPush(Qt.resolvedUrl("FirstPage.qml"))
        }
    }
}
```

```
    Component.onCompleted: {get_note_by_id(userid_cfg_value.value, current_note_id.value)}
}
// NewNote.qml
import QtQuick 2.0
import Sailfish.Silica 1.0
import Nemo.Configuration 1.0


Page {

    ConfigurationValue {
        id: userid_cfg_value
        key: "/apps/NotesMobileClient/userid"
        defaultValue: -1
    }
    ConfigurationValue {
        id: host_cfg_value
        key: "/apps/NotesMobileClient/host"
        defaultValue: ""
    }

    ConfigurationValue {
        id: current_note_id
        key: "/apps/NotesMobileClient/current_note_id"
        defaultValue: -1
    }

    function add_new_note(_userid, _title, _text) {
        const xmlHttp = new XMLHttpRequest()
        xmlHttp.onreadystatechange = function () {
            if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
                var json_resp = JSON.parse(xmlHttp.responseText)
                console.log(json_resp['success'])
            }
        }
        xmlHttp.open("POST", host_cfg_value.value+"/api/add_note", true)
        xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
        var response_json = JSON.stringify({
                                               "userid": _userid,
                                               "title": _title,
                                               "text": _text
                                           });
        xmlHttp.send(response_json)
    }



    id: page
    allowedOrientations: Orientation.All
    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height
```

```
        Column {
            id: column
            width: page.width
            spacing: Theme.paddingLarge


            Label {
                id: header
                text: "New Note"
                font.pixelSize: 50
                anchors.horizontalCenter: parent.horizontalCenter
            }


            TextArea {
                id: note_title
                text: "Title"
                background: Rectangle {
                    color: "black"
                    opacity: 0.2
                    width: parent.width
                    height: parent.height
                }
            }

            TextArea {
                id: note_text
                text: "Text"
                wrapMode: Text.WrapAnywhere
                background: Rectangle {
                    color: "black"
                    opacity: 0.2
                    width: parent.width
                    height: parent.height
                }

            }


            Row {
                x: Theme.paddingLarge
                spacing: Theme.paddingLarge
                id: row_controls
                Button {
                    id: bt_add_note
                    text: "Add"
                    onClicked: {
                        add_new_note(userid_cfg_value.value, note_title.text,
note_text.text)
                        pageStack.animatorPush(Qt.resolvedUrl("FirstPage.qml"))
                    }
```

```
                }

                Button {
                    id: bt_to_all_notes_page
                    text: "Back"
                     onClicked: pageStack.animatorPush(Qt.resolvedUrl("FirstPage.qml"))
                }

            }
        }
    }

}

// Settings.qml
import QtQuick 2.0
import Sailfish.Silica 1.0
import Nemo.Configuration 1.0


Page {

    ConfigurationValue {
        id: userid_cfg_value
        key: "/apps/NotesMobileClient/userid"
        defaultValue: -1
    }

    ConfigurationValue {
        id: host_cfg_value
        key: "/apps/NotesMobileClient/host"
        defaultValue: ""
    }

    ConfigurationValue {
        id: username_cfg_value
        key: "/apps/NotesMobileClient/username"
        defaultValue: ""
    }


    function login(_login, _password){
        const xmlHttp = new XMLHttpRequest()
        xmlHttp.onreadystatechange = function () {
            if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {
                var json_resp = JSON.parse(xmlHttp.responseText)
                var tmp = "UserId\n"
                userid_cfg_value.value = json_resp['userid']
                username_cfg_value.value = json_resp['username']
                status_label.text = "Login status: " + json_resp['success'] + ", as " +
json_resp['username']
            }
        }
        xmlHttp.open("POST", host_cfg_value.value+"/api/login", true)
```

```
        xmlHttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8")
        var response_json = JSON.stringify({"login": _login,
                                            "password": _password,
                                           });
        xmlHttp.send(response_json)
    }

    id: page

    allowedOrientations: Orientation.All


    SilicaFlickable {
        anchors.fill: parent

        contentHeight: column.height


        Column {
            id: column
            width: page.width
            spacing: Theme.paddingLarge
            property var host: "http://192.168.0.105:5000"

            PageHeader {
                Label {
                    text: "Settings"
                    font.pixelSize: 50
                    anchors.horizontalCenter: parent.horizontalCenter
                }
            }

            Label {
                text: "LOGIN SETTINGS"
                font.pixelSize: 40
            }

            Label {
                id: status_label
                text: "Login status: " + (userid_cfg_value.value === -1 ? "false" : "true,
as " + username_cfg_value.value)
            }

            TextField {
                id: login_input
                placeholderText: "login"
            }

            TextField {
                id: password_input
                placeholderText: "password"
            }

            Row {
```

```
                spacing: Theme.paddingLarge
                anchors.horizontalCenter: parent.horizontalCenter
                Button {
                    id: login_bt
                    text: "Login"
                    onClicked: {login(login_input.text, password_input.text)}
                }

                Button {
                    id: logout_bt
                    text: "Logout"
                    onClicked: {
                        userid_cfg_value.value = -1
                        username_cfg_value.value = ""
                    }
                }
            }


            Label {
                text: "HOST SETTINGS"
                font.pixelSize: 40
                y: Theme.paddingLarge
            }
            TextField {
                id: host_input
                placeholderText: "host: " + host_cfg_value.value
            }

            Button {
                id: change_host_bt
                text: "ChangeHost"
                onClicked: {host_cfg_value.value = host_input.text;
host_input.placeholderText = "host: " + host_cfg_value.value}
                anchors.horizontalCenter: parent.horizontalCenter
            }

            Button {
                id: back_to_notes_bt
                text: "Back"
                anchors.horizontalCenter: parent.horizontalCenter
                onClicked: {
                    pageStack.animatorPush(Qt.resolvedUrl("FirstPage.qml"))
                }
            }


        }
    }
}
```