ГУАП

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ст.преподаватель |  |  |  | Е.О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  К КУРСОВОМУ ПРОЕКТУ |
| Разработка приложения для организации взаимодействия объектов при заданных критериях |
| по дисциплине: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Столяров Н.С. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2023

**Содержание**

[1. Постановка задачи 3](#_Toc148454593)

[1.1 Анализ предметной области 3](#_Toc148454594)

[2 Разработка классов 4](#_Toc148454595)

[2.1 Иерархия классов 4](#_Toc148454596)

[2.2 Управляющие классы: 5](#_Toc148454597)

[2.3 Интерфейсные классы: 5](#_Toc148454598)

[2.5 Рассмотрим полученную диаграмму: 6](#_Toc148454599)

[3.1 Описание интерфейсов 6](#_Toc153289460)

[3.2 Разработка методов классов 7](#_Toc153289461)

[Тестирование программы 7](#_Toc153289462)

[Приложение 11](#_Toc153289463)

# 1. Постановка задачи

Данная курсовая работа по программированию направлена на разработку системы классов, которая будет описывать СМС уведомления. Основной целью проектирования является создание программного продукта, который позволит уведомлять пользователей о записи куда либо где будет установлено это ПО.

# 1.1 Анализ предметной области

1.1.1. Предметной областью является отправка СМС уведомлений клиентам. Основными сущностями предметной области являются клиенты, комманда, и шаблон.

1.1.2. Словарь предметной области:

* СМС - Short Message Service (служба коротких сообщений)
* USSD
* AT

1.1.3. Функциональные требования:

* Логирование всего что происходит в системе (для анализа ошибок)
* Отправка СМС уведомлений
* Связь с 1С через запросы
* Отправка USSD запросов (например для получения баланса)
* Обеспечение безопасности при общении с 1С

# 2 Разработка классов

Для разработки иерархии классов мы начнем с выделения основных сущностей предметной области и определения классов, описывающих эти сущности. Затем мы определим управляющие классы и интерфейсные классы для организации взаимодействия между ними и с внешней средой. Ниже приведена детальная разработка иерархии классов

# 2.1 Иерархия классов

2.1.1. Комманда (Command):

- command (текст комманды для отправки)  
- end (ожидаемое окончание)  
- priority (приоритет)  
Методы:  
 - Конструктор для создания

2.1.2. Клиент (User):

- phone (Номер телефона клиента)  
- checked (флаг. Ответил ли пользователь на уведомление)  
Методы:  
 - Конструктор для создания

2.1.3. Шаблон (Template):

- text(тест шаблона)  
- args (переменные для подставления)  
- default (массив значений по умолчанию)  
Методы:  
 - Конструктор для создания

# 2.2 Управляющие классы:

2.2.1. Templates (Шаблоны):

* Поля:
  + Список всех шаблонов
* Методы:
  + Методы для добавления шаблонов
  + Метод для использования шаблона (подставки текста)

2.2.2. user\_information (База клиентов):

* Поля:
  + Список всех клиентов
* Методы:
  + Методы для добавления/удаления клиентов
  + Метод для изменения статуса ответа клиента

2.2.3. SIM800L (Модуль для управления СМС):

* Поля:
  + Список всех комманд
* Методы:
  + Методы для выполнения AT комманд
  + Методы для автоматизированной отправки сообщений

# 2.2 Патерны проектирования:

**Наблюдатель (Observer):**

Паттерн наблюдатель может быть полезен для оповещения

пользователей о новых письмах, таким образом, они могут

быть уведомлены о появлении новых сообщений в реальном

времени.

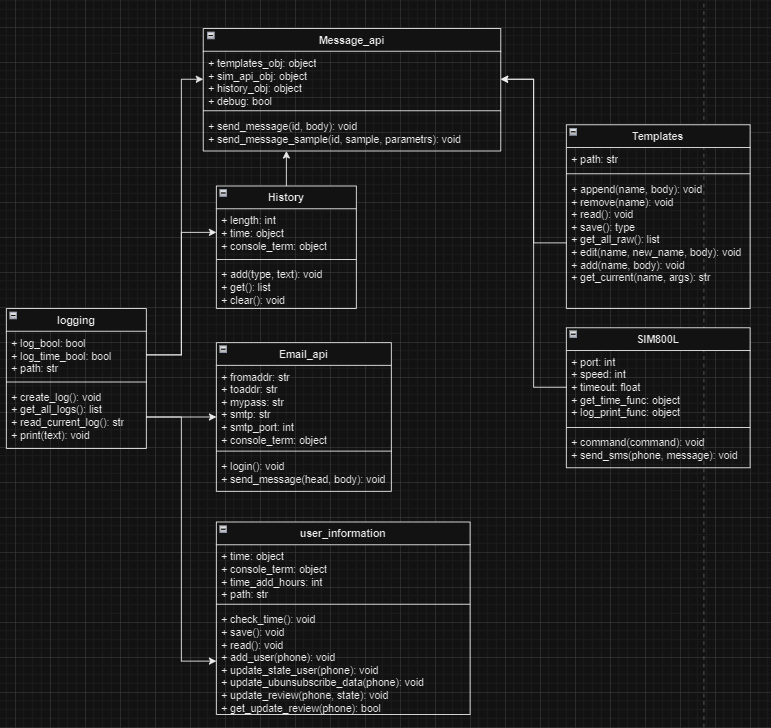
**Фабричный метод (Factory Method):**

Для создания экземпляров писем и почтовых ящиков можно

использовать фабричный метод, который позволяет

инкапсулировать процесс создания объектов.

# 2.5 Рассмотрим поfлученную диаграмму:

****

Эта иерархия классов позволит эффективно управлять данными и операциями, связанными с СМС уведомлениями, а также обеспечит удобный интерфейс для управления всей этой системой.  
  
3.1 Описание интерфейсов

Интерфейс представляет из себя набор различных пунктов меню (написанный через html). Таким образом все возможные действия можно аккуратно скомпоновать. По итогу были сделаны:  
 - Форма авторизации  
 - Различные формы для ввода сообщения (каждая форма представляет из себя шаблон сообщения)

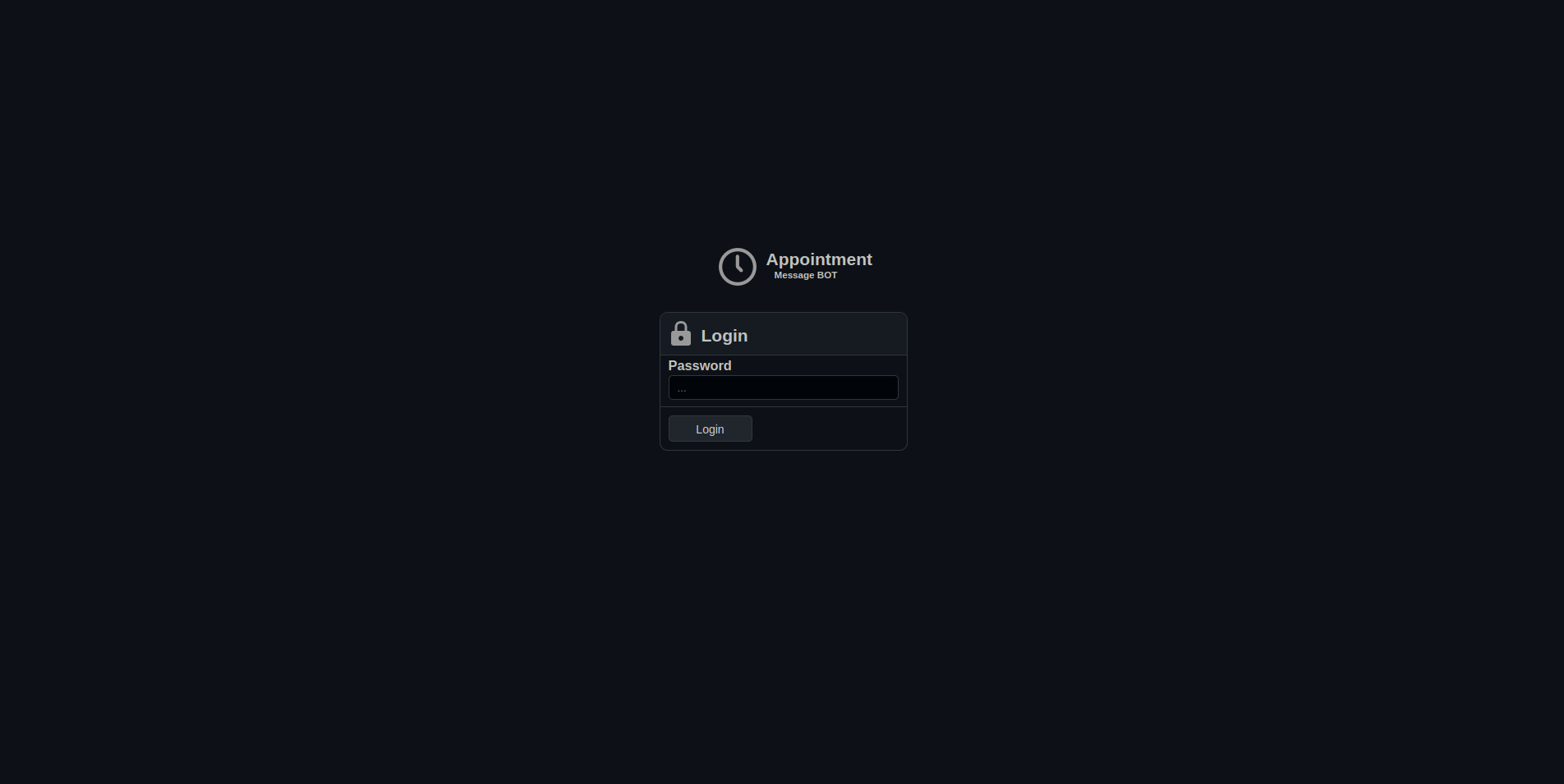
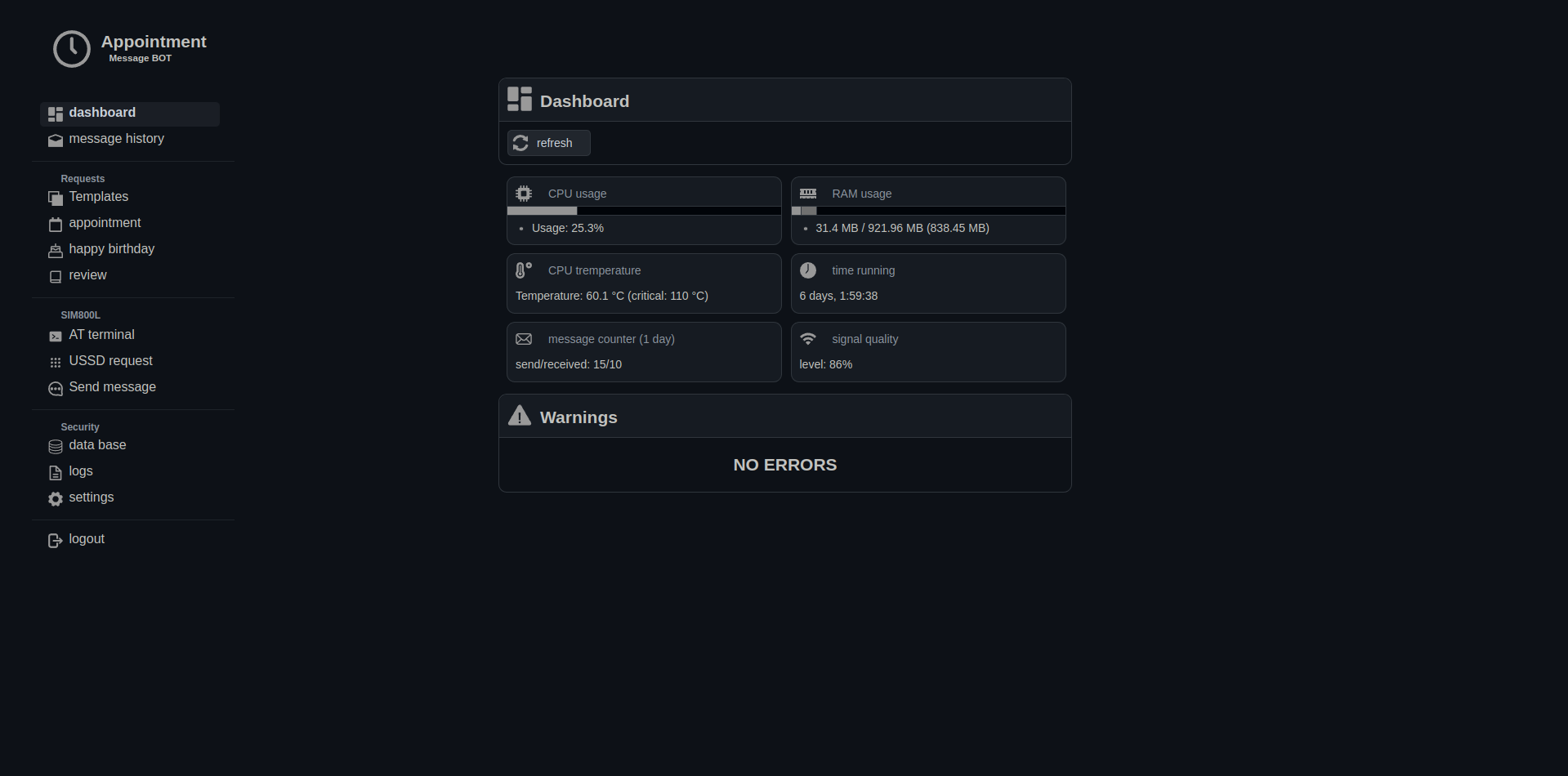
- Просмотр текущей базы с клиентами  
 - Различные формы для просмотра аудита (история сообщений, логи, AT терминал)

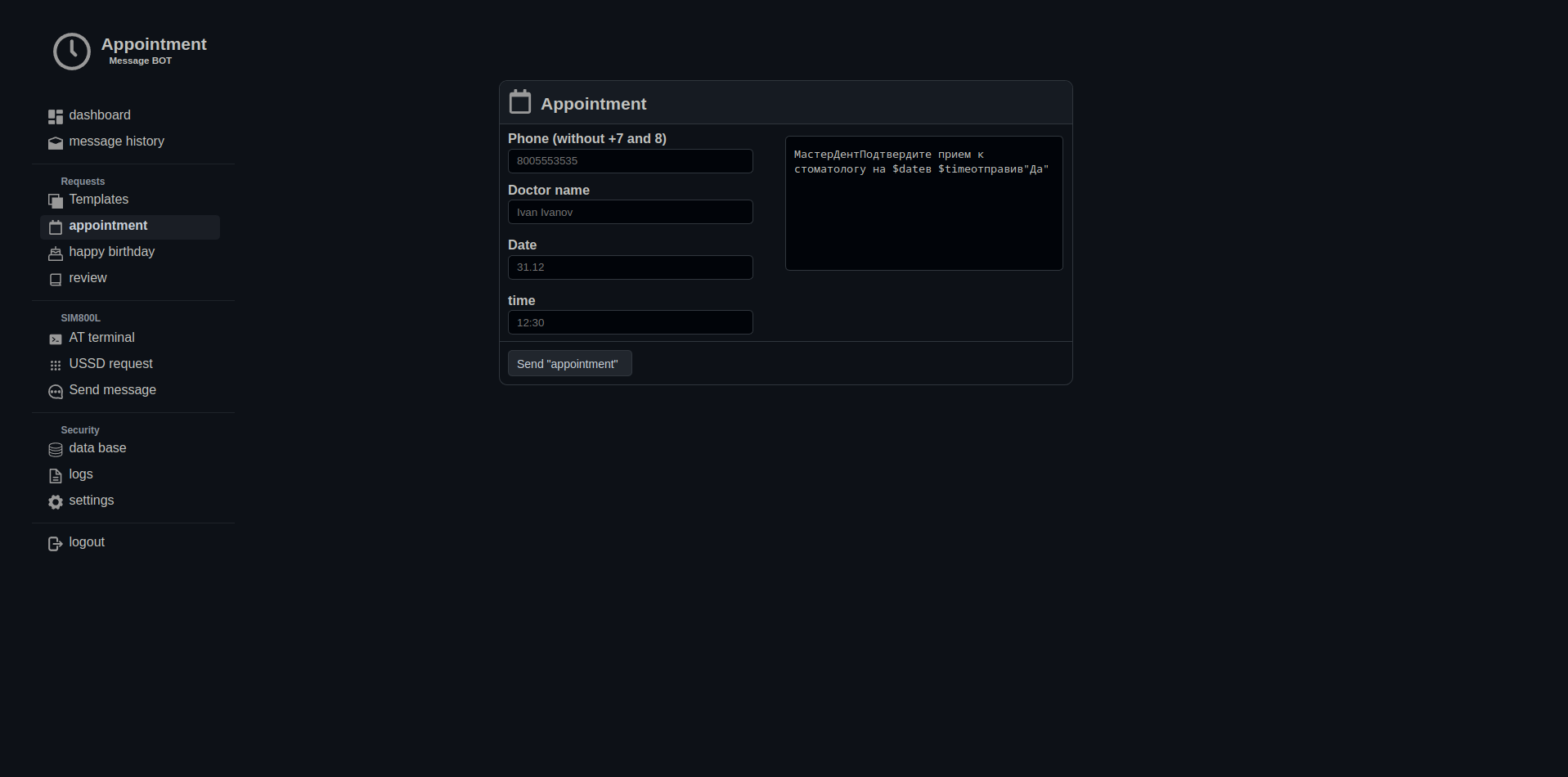
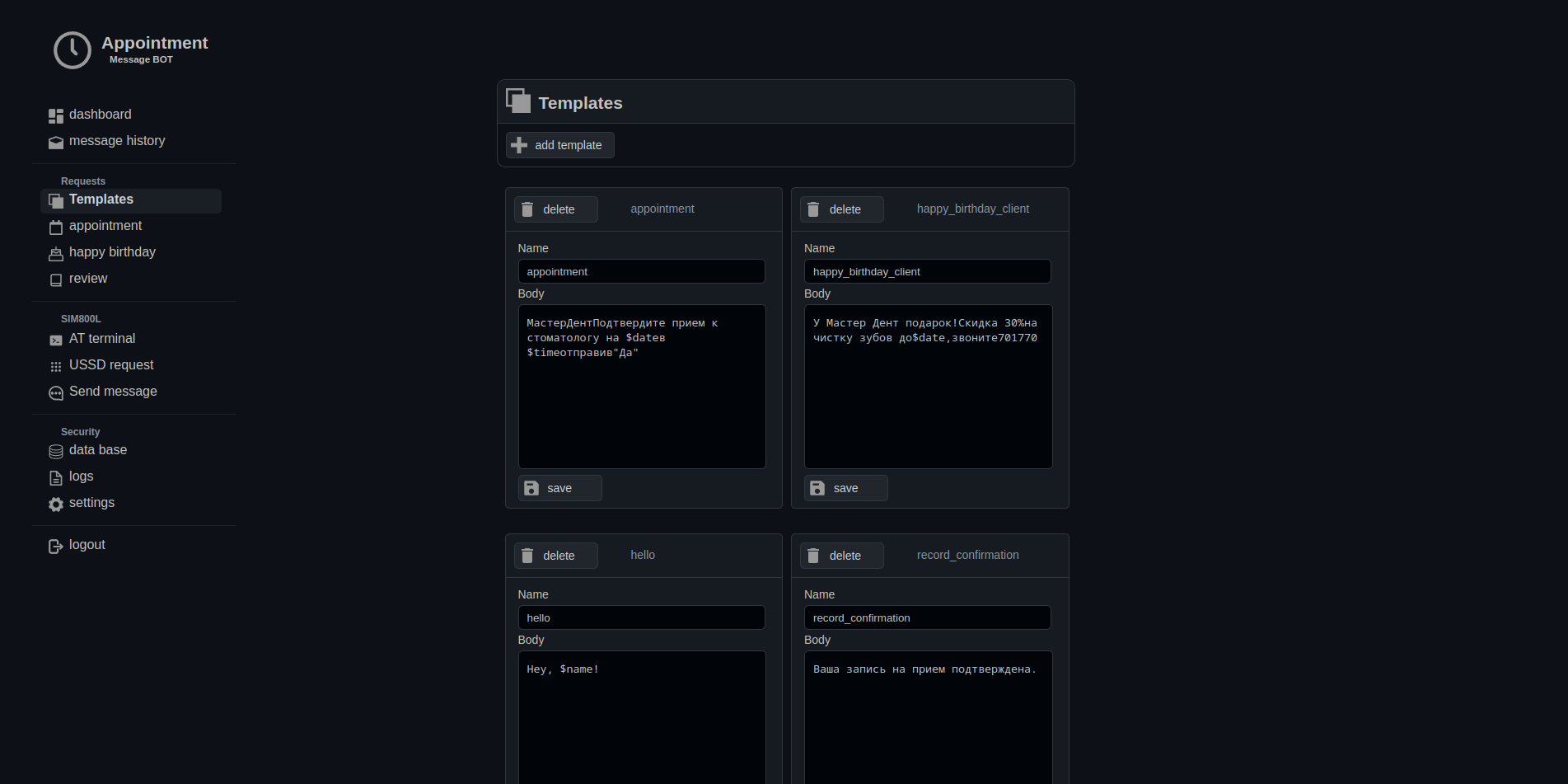
Код интерфейса

|  |
| --- |
| Авторизация |
| <!-- форма для авторизации -->  <div class=**"login\_page border"**>  <div class=**"name\_area\_border"** style=**"width: 300px; height: 51px"**>  <img class=**"icon"** width=**"30"** height=**"30"** src=**"static/img/lock.svg"**>  <h2>**Login**</h2>  </div>  <hr class=**"main\_page\_hr"**>    <!-- поле ввода пароля -->  <h3 style=**"margin: 0px 10px"**>**Password**</h3>  <input id=**"password\_input"** class=**"input\_border"** style=**"left: 10px; width: 280px; margin-top: 0px;"** type=password placeholder=**"..."** onkeypress=**"return login\_enter(event)"**>  <hr class=**"main\_page\_hr"**>  <!-- кнопка для отправки пароля -->  <div class=**"main\_page\_button"** style=**"width: 100px; margin: 10px"** onclick=**"login()"**>  <p align=**"center"**>**Login**</p>  </div>  </div> |

|  |
| --- |
| Меню |
| <!-- меню -->  <div class=**"left\_bar"**>  <!-- логотип -->  <div class=**"logo"**>  <img class=**"icon logo\_img"** width=**"50"** height=**"50"** src=**"static/img/logo.svg"**>  <h2 style=**"margin: -50px 60px;"**>**Appointment**</h2>  <h5 style=**"margin: 45px 70px;"**>**Message BOT**</h5>  </div>  <!-- список кнопок -->  <table class=**"menu\_table left\_bar\_button\_p"**>  <!-- кнопка: главная панель -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_dashboard"** onclick=**"open\_page('dashboard')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/dashboard.svg"**>  <p align=**"left"**>**dashboard**</p>  </div>  </th></tr>  <!-- кнопка: история сообщений -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_history"** onclick=**"open\_page('history')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/history.svg"**>  <p align=**"left"**>**message history**</p>  </div>  </th></tr>  <!-- линия -->  <tr><th>  <hr class=**"left\_bar\_hr"**>  </th></tr>  <!-- комментарий -->  <tr><th>  <p class=**"left\_bar\_description"** align=**"left"**>**Requests**</p>  </th></tr>  <!-- кнопка: шаблоны -->  <tr><th>  <div class=**"left\_bar\_button\_selected"** id=**"left\_bar\_templates"** onclick=**"open\_page('templates')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/template.svg"**>  <p align=**"left"**>**Templates**</p>  </div>  </th></tr>  <!-- кнопка: уведомления -->  <tr><th>  <div class=**"left\_bar\_button\_selected"** id=**"left\_bar\_appointment"** onclick=**"open\_page('appointment')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/appointment.svg"**>  <p align=**"left"**>**appointment**</p>  </div>  </th></tr>  <!-- кнопка: поздравление с днём рождения -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_happy\_birthday"** onclick=**"open\_page('happy\_birthday')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/happy birthday.svg"**>  <p align=**"left"**>**happy birthday**</p>  </div>  </th></tr>  <!-- кнопка: отзыв -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_review"** onclick=**"open\_page('review')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/review.svg"**>  <p align=**"left"**>**review**</p>  </div>  </th></tr>  <!-- линия -->  <tr><th>  <hr class=**"left\_bar\_hr"**>  </th></tr>  <!-- комментарий -->  <tr><th>  <p class=**"left\_bar\_description"** align=**"left"**>**SIM800L**</p>  </th></tr>  <tr><th>  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_at\_terminal"** onclick=**"open\_page('at\_terminal')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/terminal.svg"**>  <p align=**"left"**>**AT terminal**</p>  </div>  </th></tr>  <!-- кнопка: USSD запрос -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_ussd\_request"** onclick=**"open\_page('ussd\_request')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/ussd.svg"**>  <p align=**"left"**>**USSD request**</p>  </div>  </th></tr>  <!-- кнопка: отправить сообщение -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_send\_message"** onclick=**"open\_page('send\_message')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/message.svg"**>  <p align=**"left"**>**Send message**</p>  </div>  </th></tr>  <!-- линия -->  <tr><th>  <hr class=**"left\_bar\_hr"**>  </th></tr>  <!-- комментарий -->  <tr><th>  <p class=**"left\_bar\_description"** align=**"left"**>**Security**</p>  </th></tr>  <!-- кнопка: база данных -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_data\_base"** onclick=**"open\_page('data\_base')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/data base.svg"**>  <p align=**"left"**>**data base**</p>  </div>  </th></tr>  <!-- кнопка: логи -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_logs"** onclick=**"open\_page('logs')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/log.svg"**>  <p align=**"left"**>**logs**</p>  </div>  </th></tr>  <!-- кнопка: настройки -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_settings"** onclick=**"open\_page('settings')"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/settings.svg"**>  <p align=**"left"**>**settings**</p>  </div>  </th></tr>  <!-- линия -->  <tr><th>  <hr class=**"left\_bar\_hr"**>  </th></tr>  <!-- кнопка: выход -->  <tr><th>  <div class=**"left\_bar\_button"** id=**"left\_bar\_data\_base"** onclick=**"logout()"**>  <img class=**"icon"** width=**"18"** height=**"18"** src=**"static/img/logout.svg"**>  <p align=**"left"**>**logout**</p>  </div>  </th></tr>  </table>  </div> |

3.2 Разработка методов классов  
  
  
  
Тестирование программы  
Страница авторизации

  
Главная панель  
  
  
  
  
  
  
Отправка сообщения с шаблоном (appointment)

  
Редактор шаблонов  
  
Приложение

|  |
| --- |
| sim\_api.py |
| # модуль для работы с Sim800L через arduino  #  # пример инициализации класса:  # sms\_API = Sim800l("/dev/ttyUSB0", 9600)  #  # пример отправки сообщения:  # sms\_API.read\_messages()  #  # пример кода для чтения сообщений:  # while True:  # if len(sms\_API.messages) > 0:  # for el in sms\_API.messages:  # print(el)  # sms\_API.send\_sms(el["phone"], el["body"])  # sms\_API.messages = []  **import** time  **import** serial  **import** binascii  **import** threading  REMOVE\_SYMBOLS **=** **[**"\n"**,** "\r"**]**  AT\_HISTORY\_LENGTH **=** 100  MAX\_USSD\_RESULT **=** 5  **class** **SIM800L:**  **def** \_\_init\_\_**(**self**,** port**,** speed**=**9600**,** timeout**=**0.1**,** get\_time\_func**=None,** log\_print\_func**=None):**  self**.**commands\_queue **=** **[]**  self**.**command\_end\_flag **=** **False**  self**.**messages **=** **[]**  self**.**ussd\_result **=** **[]**  self**.**signal\_quality **=** 100  self**.**check\_signal\_flag **=** **False**  self**.**signal\_quality\_delay **=** 60  self**.**signal\_quality\_time **=** time**.**perf\_counter**()** **+** self**.**signal\_quality\_delay  self**.**at\_history **=** **[]**  self**.**get\_time\_func **=** get\_time\_func  self**.**terminal\_io **=** **None**  self**.**log\_print\_func **=** log\_print\_func  self**.**timeout **=** timeout  self**.**ser **=** serial**.**Serial**(**  port**=**port**,**  baudrate **=** speed**,**  parity**=**serial**.**PARITY\_NONE**,**  stopbits**=**serial**.**STOPBITS\_ONE**,**  bytesize**=**serial**.**EIGHTBITS**,**  timeout**=**timeout  **)**  self**.**read\_thread **=** threading**.**Thread**(**target**=**self**.**read\_thread\_func**)**  self**.**read\_thread**.**start**()**  self**.**write\_thread **=** threading**.**Thread**(**target**=**self**.**write\_thread\_func**)**  self.write\_thread.start()  self.configure()  self.read\_messages()  def append\_history(self, text):  for el in REMOVE\_SYMBOLS:  text = text.replace(el, "")  self.at\_history.append(text)  while len(self.at\_history) > AT\_HISTORY\_LENGTH:  self.at\_history.pop(0)  self.log\_print\_func.print(text, 0)  if self.terminal\_io != None:  self.terminal\_io(text)  def command(self, command, encode=True):  self.commands\_queue.append(  (str(command).encode() if encode else command)  )  def read\_thread\_func(self):  buf = {}  while True:  try:  if self.signal\_quality\_time <= time.perf\_counter() and not self.check\_signal\_flag:  self.check\_signal()  out = self.ser.readline()  try:  out = out.decode()  except:  pass  if len(out) > 0:  for el in REMOVE\_SYMBOLS:  out = out.replace(el, "")  if (out in ["START", "OK", "ERROR", "> "]):  self.command\_end\_flag = True  if len(out) > 0:  self.append\_history("[%s] >> %s" % (self.get\_time\_func(), out))  out\_splited = out.split(":")  if out\_splited[0] == "+CMTI":  self.read\_messages()  elif out\_splited[0] == "+CUSD":  body = out\_splited[1].split("\"")[1]  try:  self.ussd\_result.append(binascii.unhexlify(body).decode("utf-16-be"))  except:  self.ussd\_result.append(binascii.unhexlify(body).decode())  while len(self.ussd\_result) > MAX\_USSD\_RESULT:  self.ussd\_result.pop(0)  elif out\_splited[0] == "+CMGL":  buf = {  "phone": binascii.unhexlify(out\_splited[1].split(",")[2].replace("\"", "")).decode()  }  if out\_splited[0] == "+CSQ":  self.signal\_quality = int(out\_splited[1].split(',')[0])  self.signal\_quality\_time = time.perf\_counter() + self.signal\_quality\_delay  self.check\_signal\_flag = False  elif len(buf) > 0:  try:  buf["body"] = binascii.unhexlify(out).decode("utf-16-be")  except:  buf["body"] = binascii.unhexlify(out).decode()  self.messages.append(buf.copy())  buf = {}  except Exception as e:  print("Serial READ THREAD", e)  def write\_thread\_func(self):  while True:  if len(self.commands\_queue) and self.command\_end\_flag > 0:  self.append\_history("[%s] << %s" % (self.get\_time\_func(), self.commands\_queue[0].decode()))  self.ser.write(self.commands\_queue[0])  self.commands\_queue.pop(0)  self.command\_end\_flag = False  else:  time.sleep(self.timeout)  def reboot(self):  self.command("AT+CFUN=1,1\r\n")  def configure(self):  self.command("ATZ\r\n")  self.command("AT+CSCS=\"HEX\"\r\n")  self.command("AT+CSMP=17,168,0,8\r\n")  self.command("AT+CMGF=1\r\n")  self.command("AT+GSMBUSY=1\r\n")  self.check\_signal()  def check\_signal(self):  self.command("AT+CSQ\r\n")  self.check\_signal\_flag = True  def send\_sms(self, phone, message):  self.command("AT+CMGS=\"%s\"\r\n" % phone)  self.command(binascii.hexlify(str(message).encode('utf-16-be')) + chr(26).encode(), encode=False)  self.delete\_send()  def send\_ussd(self, code):  self.command("AT+CUSD=1,\"%s\"\r\n" % code)  def read\_messages(self):  self.command("at+cmgl=\"REC UNREAD\"\r\n")  self.delete\_read()  def delete\_send(self):  self.command("AT+CMGDA=\"DEL SENT\"\r\n")  def delete\_read(self):  self.command("AT+CMGDA=\"DEL READ\"\r\n") |

|  |
| --- |
| user\_information.py |
| **import** os  **import** json  **import** threading  **import** time  **from** app**.**dict\_json **import** **\***  # типа БД для хранения данных о номерах  **class** **user\_information():**  **def** \_\_init\_\_**(**self**,** time**,** console\_term**,** time\_add\_hours**=**36**,** path**=**'base'**):**  self**.**time **=** time  self**.**console\_term **=** console\_term  self**.**path **=** 'base'  self**.**error\_promt **=** 'USER INFORMATION: '  self**.**data **=** **{}**  self**.**unsubscribe\_data **=** **{}**  self**.**review **=** **{}**  self**.**read**()**  self**.**run\_check **=** **True**  self**.**time\_add\_hours **=** time\_add\_hours  self**.**thread **=** threading**.**Thread**(**target**=**self**.**check\_time**)**  self**.**thread**.**daemon **=** **True**  #self.thread.start()  **def** check\_time**(**self**):**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Thread start"**,** 0**)**  **while** self**.**run\_check**:**  edited **=** **False**  **try:**  ################ Уведомления о приёме  delete\_phone\_list **=** **[]**  **for** phone **in** self**.**data**:**  **if** **(not** self**.**data**[**phone**][**0**]** **and** self**.**time**.**check\_data\_for\_base**(**self**.**data**[**phone**][**1**])):**  delete\_phone\_list**.**append**(**phone**)**  **for** phone **in** delete\_phone\_list**:**  self**.**data**.**pop**(**phone**)**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Deleted phone " **+** phone**,** 0**)**  edited **=** **True**  ################# Уведомления о отзывах и их стадиях  delete\_phone\_list **=** **[]**  **for** phone **in** self**.**review**:**  **if** **(**self**.**time**.**check\_data\_for\_base**(**self**.**review**[**phone**][**1**])):**  delete\_phone\_list**.**append**(**phone**)**  **for** phone **in** delete\_phone\_list**:**  self**.**review**.**pop**(**phone**)**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Deleted phone review " **+** phone**,** 0**)**  edited **=** **True**  # сохраняем  **if** edited**:**  self**.**save**()**  **except** Exception **as** e**:**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Thread => " **+** str**(**e**),** 3**)**  time**.**sleep**(**1**)**  **def** save**(**self**):**  data **=** **{**  'confirm'**:** self**.**data**,**  'unsubscribe'**:** self**.**unsubscribe\_data**,**  'review'**:** self**.**review  **}**  save\_dict**(**data**,** self**.**path**)**  **def** read**(**self**):**  **if** **not** os**.**path**.**exists**(**self**.**path **+** '.json'**):**  self**.**save**()**  **else:**  **try:**  data **=** read\_dict**(**self**.**path**)**  self**.**data **=** data**[**'confirm'**]**  self**.**unsubscribe\_data **=** data**[**'unsubscribe'**]**  self**.**review **=** data**[**'review'**]**  **except:**  self**.**save**()**  # добавление номера в базу  **def** add\_user**(**self**,** phone**):**  self**.**data**[**phone**]** **=** **[**  **False,**  self**.**time**.**get\_data\_for\_base**(**self**.**time\_add\_hours**)**  **]**  self**.**save**()**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Add phone(confirm) " **+** phone**,** 0**)**  # обновление состояния записи у номера  **def** update\_state\_user**(**self**,** phone**):**  **if** phone **in** self**.**data**:**  self**.**data**[**phone**][**0**]** **=** **True**  self**.**save**()**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Update phone(confirm) " **+** phone**,** 0**)**  # добавление номера в базу отписавшихся от расслок  **def** update\_ubunsubscribe\_data**(**self**,** phone**):**  self**.**unsubscribe\_data**[**phone**]** **=** **True**  self**.**save**()**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Add phone(unsubscribe) " **+** phone**,** 0**)**  **def** update\_review**(**self**,** phone**,** state**=**0**):**  # state:  # 0 - предложить пользователю оценить от 0 до 5  # 1 - дать пользователю ввести отзыв в сообщении  # -1 - удалить из списка  **if** state **==** **-**1**:**  **if** phone **in** self**.**review**:**  self**.**review**.**pop**(**phone**)**  **elif** state **in** **[**0**,** 1**]:**  self**.**review**[**phone**]** **=** **[**  state**,**  self**.**time**.**get\_data\_for\_base**(**self**.**time\_add\_hours**)**  **]**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Update state(" **+** str**(**state**)** **+** ") " **+** phone**,** 0**)**  **else:**  self**.**console\_term**.print(**str**(**self**.**error\_promt**)** **+** "Error state(" **+** str**(**state**)** **+** ") " **+** phone**,** 3**)**  self**.**save**()**  **def** get\_update\_review**(**self**,** phone**):**  **if** phone **in** self**.**review**:**  **return** self**.**review**[**phone**][**0**]**  **else:**  **return** **False** |