

2 grudnia 2018

Piotr Cieszko 226079
Wojciech Ormaniec 226181

grupa:
czwartek TP, 8¹⁵-11⁰⁰

Internetowe Bazy Danych

System wspierający działanie firmy w warstwie informacyjnej

Prowadzący:
dr inż. Roman Ptak

Rok akademicki:
2018/2019

Spis treści

1	Założenia projektowe	3
1.1	Przypadki użycia	3
1.2	Bezpieczeństwo	5
1.3	Przewidywane technologie	5
2	Analiza SWOT	6
2.1	Analiza powiązań SWOT	7
3	Techniczna wykonalność projektu	9
4	Kosztorys	9
5	Instrukcja dla użytkownika	10
5.1	Rejestracja w systemie	10
5.2	Logowanie do systemu	11
5.3	Wylogowanie z systemu	11
5.4	Sprawdzenie informacji o sobie	12
5.5	Odczytywanie notyfikacji	12
5.6	Sprawdzenie listy wszystkich użytkowników systemu (tylko dla administratora)	13
5.7	Sprawdzenie listy faktur i pobranie interesujących	13
5.8	Wystawienie nowych faktur (dla pracownika)	14
5.9	Porzędkanie forum	14
5.10	Dodawanie nowych wątków i postów	15
6	Opis aplikacji od strony programistycznej	17
6.1	Architektura aplikacji	17
6.2	Bezpieczeństwo	17
6.3	Diagram związków encji	18
6.4	Mapowanie ORM	19
6.5	Widoki	19
6.6	Szablony	21
6.7	Urls	23
7	Instrukcja wdrożenia	24
8	Podsumowanie	25
9	Literatura	25

1 Założenia projektowe

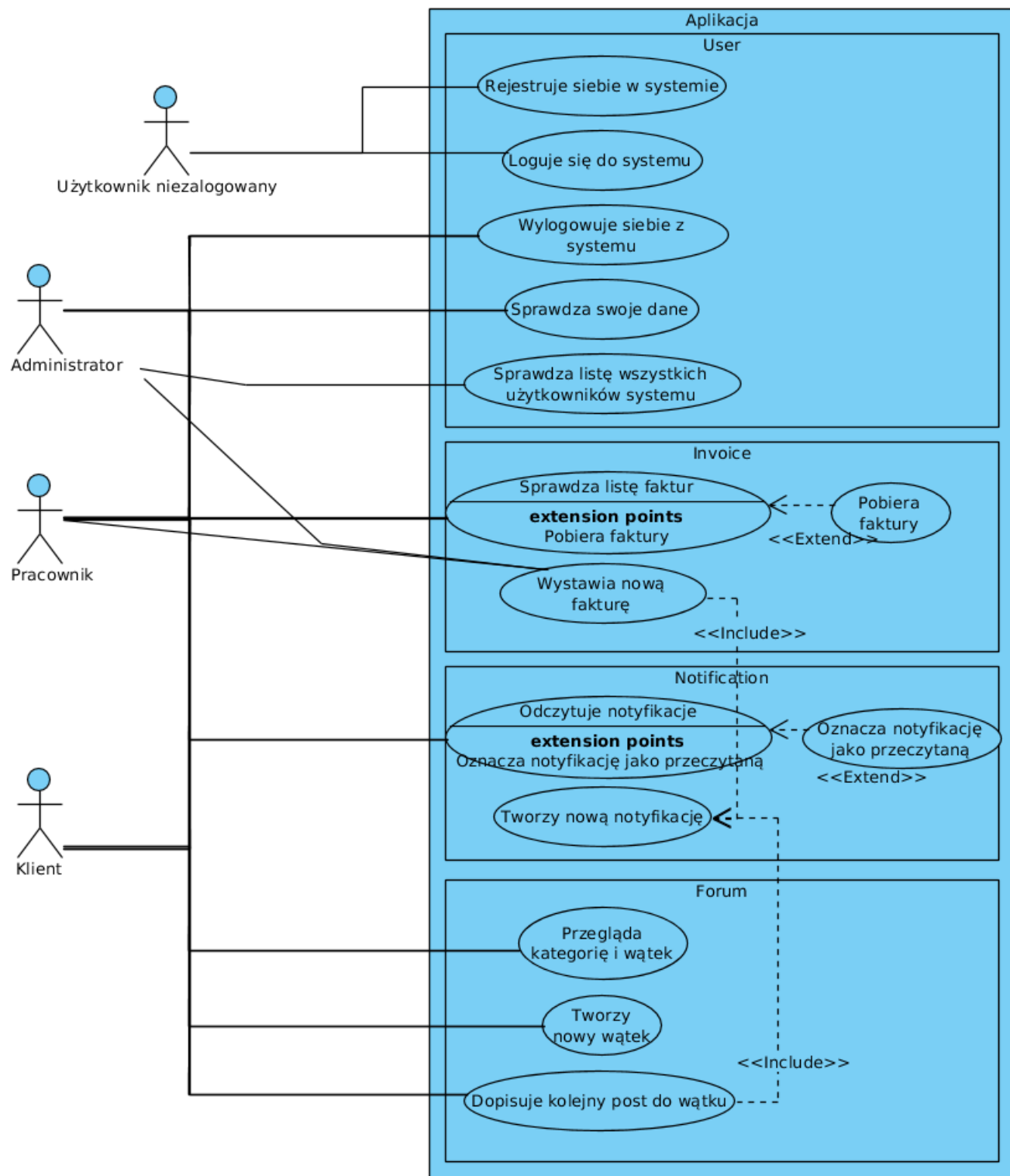
Celem projektu jest stworzenie aplikacji internetowej, która będzie realizować określone funkcjonalności w firmie klienta z uwzględnieniem łatwości dalszej rozbudowy.

Podstawą tej aplikacji będzie system zarządzania pracownikami oraz klientami firmy - zarządzanie dostępem, podstawowymi danymi personalnymi oraz udostępnionymi funkcjami.

- Pierwszą funkcjonalnością będzie zarządzanie fakturami, które wystawiamy klientom. Klient, któremu pracownik nadał dostęp do tych zasobów, może samodzielnie pobierać udostępnione faktury. Pracownicy z kolei mogą zarządzać fakturami w systemie, wraz z przypisaniem do określonego użytkownika.
- Drugą rzeczą do zaimplementowania jest system powiadomień. Klient może otrzymać powiadomienie wysłane przez pracownika do niego, bądź do wszystkich klientów.
- Ostatnią będzie proste forum. Możliwość tworzenia wątków, udzielania się w nich, jeden poziom kategorii tematycznych. Notyfikacje o pojawieniu się nowych postów.

1.1 Przypadki użycia

Wspólne przypadki użycia dla administratora, pracownika i klienta zostały oznaczone wspólną linią, pokolorowaną dodatkowo kolorem czerwonym dla czytelności.



Rysunek 1: Diagram przypadków użycia

1.2 Bezpieczeństwo

Strona będzie zabezpieczona przed nieuprawnionym dostępem poprzez system logowania. W przypadku dostępu do strony bez należytych uprawnień, zostanie wyświetlony komunikat o błędzie.

1.3 Przewidywane technologie

Po stronie klienta:

- HTML 4

Po stronie serwera:

- Linux
- Python
- Django
- SQLite

2 Analiza SWOT

Waga	Czynniki wewnętrzne	Waga	Czynniki zewnętrzne
1,00	Mocne strony	1,00	Szanse
0,10	Możliwość modułowego rozwoju aplikacji.	0,15	Możliwość zmniejszenia obciążenia pracowników przedsiębiorcy.
0,20	Popularne technologie ułatwiają proces tworzenia, wdrożenia oraz utrzymania aplikacji.	0,30	Nasza aplikacja poprzez swoją prostotę może zainteresować więcej klientów.
0,15	Nawet przy oferowaniu niskich cen będziemy w stanie zreważnować to wielkością sprzedaży.	0,15	Nie ma żadnych zagrożeń ze strony prawa - nie łamiemy żadnych praw
0,25	Łatwa testowalność (dzięki popularnym technologiom i prostym rozwiązaniom).	0,40	Utrzymanie klienta (firmy) przez dłuższy czas poprzez dostarczanie zintegrowanego systemu, z możliwością elastycznej rozbudowy.
0,10	Nie wymaga specjalistycznego sprzętu.		
0,20	Niewielki koszt wykonania		
1,00	Słabe strony	1,00	Zagrożenia
0,50	Krótki czas realizacji projektu	0,15	Potencjalna konieczność zatrudnienia dodatkowej osoby do obsługi programu.
0,25	Brak doświadczenia w web developmencie	0,25	Nasycony rynek.
0,25	Brak funduszy	0,25	Wyspecjalizowana funkcjonalność finalnego produktu dla poszczególnych firm.
		0,25	Ciągły support aplikacji
		0,10	Potencjalne ataki hakerskie/crackerskie

Rysunek 2: Czynniki analizy SWOT

2.1 Analiza powiązań SWOT

Mocne strony/ Szanse	[S1]	[S2]	[S3]	[S4]	[S5]	[S6]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[O1]	0	0	0	0	1	0	0,15	1	0,15	3
[O2]	1	2	2	1	0	1	0,30	7	2,10	1
[O3]	0	0	0	0	0	0	0,15	0	0,00	4
[O4]	2	1	0	1	0	1	0,40	5	2	2
Waga	0,10	0,20	0,15	0,25	0,10	0,20				
Liczba interakcji	3	3	2	2	1	2				
Iloczyn wag i interakcji	0,30	0,60	0,30	0,50	0,10	0,40				
Ranga	4	1	4	2	5	3				
Suma interakcji								26		
Suma iloczynów									6.45	

Rysunek 3: Czy określona mocna strona pozwala wykorzystać daną szansę?

Zagrożenie/ Mocne strony	[T1]	[T2]	[T3]	[T4]	[T5]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[S1]	0	1	2	0	0	0,10	3	0,30	4
[S2]	0	0	0	2	1	0,20	3	0,60	7
[S3]	0	2	1	1	0	0,15	4	0,60	4
[S4]	0	0	1	2	1	0,25	3	0,75	1
[S5]	1	0	0	1	0	0,10	2	0,20	2
[S6]	0	1	2	2	0	0,20	5	1,00	
Waga	0,15	0,25	0,25	0,25	0,10				
Liczba interakcji	1	4	6	8	2				
Iloczyn wag i interakcji	0,15	1,00	1,50	2,00	0,20				
Ranga	5	3	2	1	4				
Suma interakcji							41		
Suma iloczynów								8.3	

Rysunek 4: Czy określona mocna strona pozwala ograniczyć dane zagrożenie?

Szanse/ Słabe strony	[O1]	[O2]	[O3]	[O4]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[W1]	0	1	0	1	0,50	2	1,00	2
[W2]	0	1	0	1	0,25	2	0,50	3
[W3]	0	1	2	2	0,25	5	1,25	1
Waga	0,15	0,30	0,15	0,40				
Liczba interakcji	0	3	2	4				
Iloczyn wag i interakcji	0,00	0,90	0,30	0,80				
Ranga	4	1	3	2				
Suma interakcji						18		
Suma iloczynów							4,75	

Rysunek 5: Czy określona słaba strona ogranicza możliwość wykorzystania danej szansy?

Zagrożenie/ Słabe strony	[T1]	[T2]	[T3]	[T4]	[T5]	Waga	Liczba interakcji	Iloczyn wag i interakcji	Ranga
[W1]	0	0	1	0	1	0,50	2	1,00	1
[W2]	0	0	1	0	1	0,25	2	0,50	3
[W3]	0	1	1	1	0	0,25	3	0,75	2
Waga	0,15	0,25	0,25	0,25	0,10				
Liczba interakcji	0	1	3	1	2				
Iloczyn wag i interakcji	0,00	0,25	0,75	0,25	0,20				
Ranga	4	2	1	2	3				
Suma interakcji							14		
Suma iloczynów								3,70	

Rysunek 6: Czy określona słaba strona potęguje dane zagrożenie?

Z przeprowadzonej analizy wynika, że korzystna będzie zarówno strategia konserwatywna (8.30), jak i agresywna (6.45).

3 Techniczna wykonalność projektu

Do wykonania projektu porzebujemy:

- Czas i umiejętności potrzebne do:
 - stworzenia aplikacji (programiści, projektanci, etc.)
 - utrzymania wdrożenia u klienta
- Sprzęt:
 - komputery do tworzenia aplikacji
 - serwer do hostowania aplikacji u klienta
- Oprogramowanie:
 - Python 3.6
 - Django 2.1.2
 - Baza danych (SQLite 3)
 - System operacyjny (Linux)
 - Przeglądarka internetowa (Firefox, inne)
 - Środowisko programistyczne (PyCharm)

4 Kosztorys

- Tworzenie aplikacji i wdrożenie: $60h * 40zł = 2400zł$
- Utrzymanie wdrożenia u klienta (1 rok): $20h * 40zł = 800zł$
- Komputery – prywatne (koszt amortyzacji): 40zł
- Serwer wirtualny (1 rok): 360zł (posiadający Python 3.6)
- Domena: 150zł
- Python: 0zł
- Django: 0zł
- SQLite: 0zł
- Linux: 0 zł
- Firefox: 0 zł
- PyCharm: 0 zł

Suma (wykonanie + 1 rok utrzymania): **3750zł**

5 Instrukcja dla użytkownika

5.1 Rejestracja w systemie

Po otwarciu strony widzimy stronę główną:

Welcome on the index page !

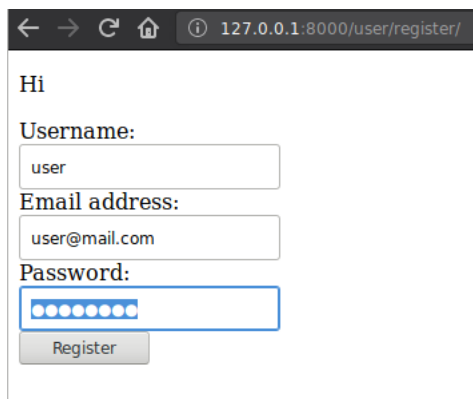
We might not be skilled with CCS but we do make solid things. Or so we think.

Do you want to login or register?

[Login](#) [Register](#)

Rysunek 7: Strona główna

Po wciśnięciu przycisku 'Register' przechodzimy do stosownego formularza:



Hi

Username:

Email address:

Password:

Rysunek 8: Formularz rejestracji

Po wypełnieniu klikamy 'Register' i zostajemy automatycznie zalogowani na nasze nowe konto. W pierwszym wierszu widzimy nazwę naszego konta ('user')

Welcome on the index page user!

We might not be skilled with CCS but we do make solid things. Or so we think.

You logged in successfully!

Notifications : 0

Places you might want to visit :

- [User](#)
- [Invoices](#)
- [Forum](#)

[logout](#)

Rysunek 9: Strona po zalogowaniu

5.2 Logowanie do systemu

Na stronie głównej wybieramy przycisk 'Login':

Welcome on the index page !

We might not be skilled with CCS but we do make solid things. Or so we think.

Do you want to login or register?

[Login](#) [Register](#)

Rysunek 10: Strona główna

Po czym wprowadzamy nasze dane i zatwierdzamy:



← → ↻ 🏠 ⓘ 127.0.0.1:8000/user/login/

Username:

Password:

Login

Rysunek 11: Logowanie

5.3 Wylogowanie z systemu

Aby się wylogować, naciskamy przycisk logout na stronie głównej.

Welcome on the index page user!

We might not be skilled with CCS but we do make solid things. Or so we think.

You logged in successfully!

Notifications : 0

Places you might want to visit :

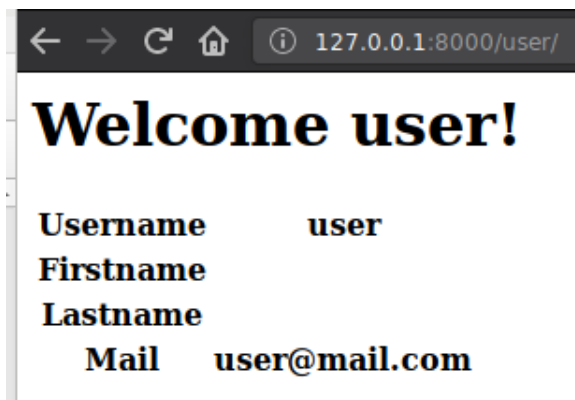
- [User](#)
- [Invoices](#)
- [Forum](#)

[logout](#)

Rysunek 12: Strona po zalogowaniu

5.4 Sprawdzenie informacji o sobie

Na stronie głównej wybieramy 'User'



Rysunek 13: Informacje o użytkowniku

5.5 Odczytywanie notyfikacji

Gdy pojawi się nowy komunikat, na stronie głównej będzie to widać w odpowiedniej sekcji:

Notifications : 1

- New incoice! [Ok](#)

Rysunek 14: Informacja o nowej fakturze

Aby schować komunikat, należy kliknąć 'Ok' przy informacji.

5.6 Sprawdzenie listy wszystkich użytkowników systemu (tylko dla administratora)

Administrator po kliknięciu 'User' z panelu głównego, ma dostęp do dodatkowego raportu po kliknięciu na link 'Go to list':



Rysunek 15: Informacje o userze dla administratora

Welcome admin!

Username admin
Firstname
Lastname
Mail
[Go to list](#)

Rysunek 16: Lista użytkowników

5.7 Sprawdzenie listy faktur i pobranie interesujących

Z panelu głównego wybieramy 'Invoices':

Here are invoices connected to your profile :

Issuinguser: Receivinguser: Invoice: No file selected.

Issuer	Receiver	Date	Name	Invoice
pracownik	user	Dec. 3, 2018	faktura.pdf	Preview

[back](#)

Rysunek 17: Lista faktur

Widzimy wszystkie adresowane do nas faktury. Po wybraniu 'Preview' możemy pobrać wybraną.

5.8 Wystawienie nowych faktur (dla pracownika)

Z panelu głównego wybieramy 'Invoices':

Here are invoices connected to your profile :

Issuinguser: Receivinguser: Invoice: faktura.pdf

Issuer	Receiver	
--------	----------	--

[back](#)

Rysunek 18: Pusta lista faktur, podczas wypełniania

Należy wypełnić pola, aby dodać fakturę. Po dodaniu faktury automatycznie tworzona jest notyfikacja.

Here are invoices connected to your profile :

Issuinguser: Receivinguser: Invoice: No file selected.

Issuer	Receiver	Date	Name	Invoice
pracownik	user	Dec. 3, 2018	faktura.pdf	Preview

[back](#)

Rysunek 19: Dodana faktura

5.9 Przeglądanie forum

Po wybraniu 'Forum' ze strony głównej, możemy przeglądać kategorie, oraz dodawać nowe:

Forum

Categories list:

Name:

- [Sprawy bieżące](#)
- [Regulaminy](#)

[home](#)

Rysunek 20: Widok kategorii

Tak samo po wejściu w kategorię widzimy poszczególne wątki:

Threads list:

Subject:

- [Przerwa w dostawie prądu 23.11.2018](#)

[back home](#)

Rysunek 21: Widok wątków

5.10 Dodawanie nowych wątków i postów

We właściwym widoku dodajemy nowy wątek (automatycznie jest generowany pusty)

Threads list:

Subject:

- [Przerwa w dostawie prądu 23.11.2018](#)

[back home](#)

Rysunek 22: Widok wątków

Potem wchodzimy w niego i możemy już swobodnie dodawać posty:

Przerwa w dostawie prądu 23.11.2018

pracownik, Dec. 2, 2018, 7:11 a.m.

Jutro o godzinie 14 zapowiedziana jest przerwa w dostawie prądu. Przewidywany czas to 3 godziny.

pracownik, Dec. 2, 2018, 7:12 a.m.

Możliwe jest przesunięcie się terminu na późniejsze godziny

Content:

[back to category home](#)

Rysunek 23: Widok pojedynczego wątku

Dodanie postu generuje notyfikację:

Notifications : 3

- New post in Przerwa w dostawie prądu 23.11.2018 [Ok](#)
- New post in Przerwa w dostawie prądu 23.11.2018 [Ok](#)
- New post in Przerwa w dostawie prądu 23.11.2018 [Ok](#)

Rysunek 24: Lista notyfikacji

6 Opis aplikacji od strony programistycznej

6.1 Architektura aplikacji

W celu utrzymania prostoty projektu, użyto standardowego dla Django wzorca model-template-view, który jest pokrewny z MVC. Dzięki temu mamy wydzieloną część bazodanową, ukrytą za modelem, szablony poszczególnych podstron oraz obsługę widoków. Każdy z tych elementów zostanie przedstawiony poniżej.

6.2 Bezpieczeństwo

Podstrony z ograniczonym dostępem dla danych użytkowników są dodatkowo sprawdzane podczas wczytywania (pierwsza linia):

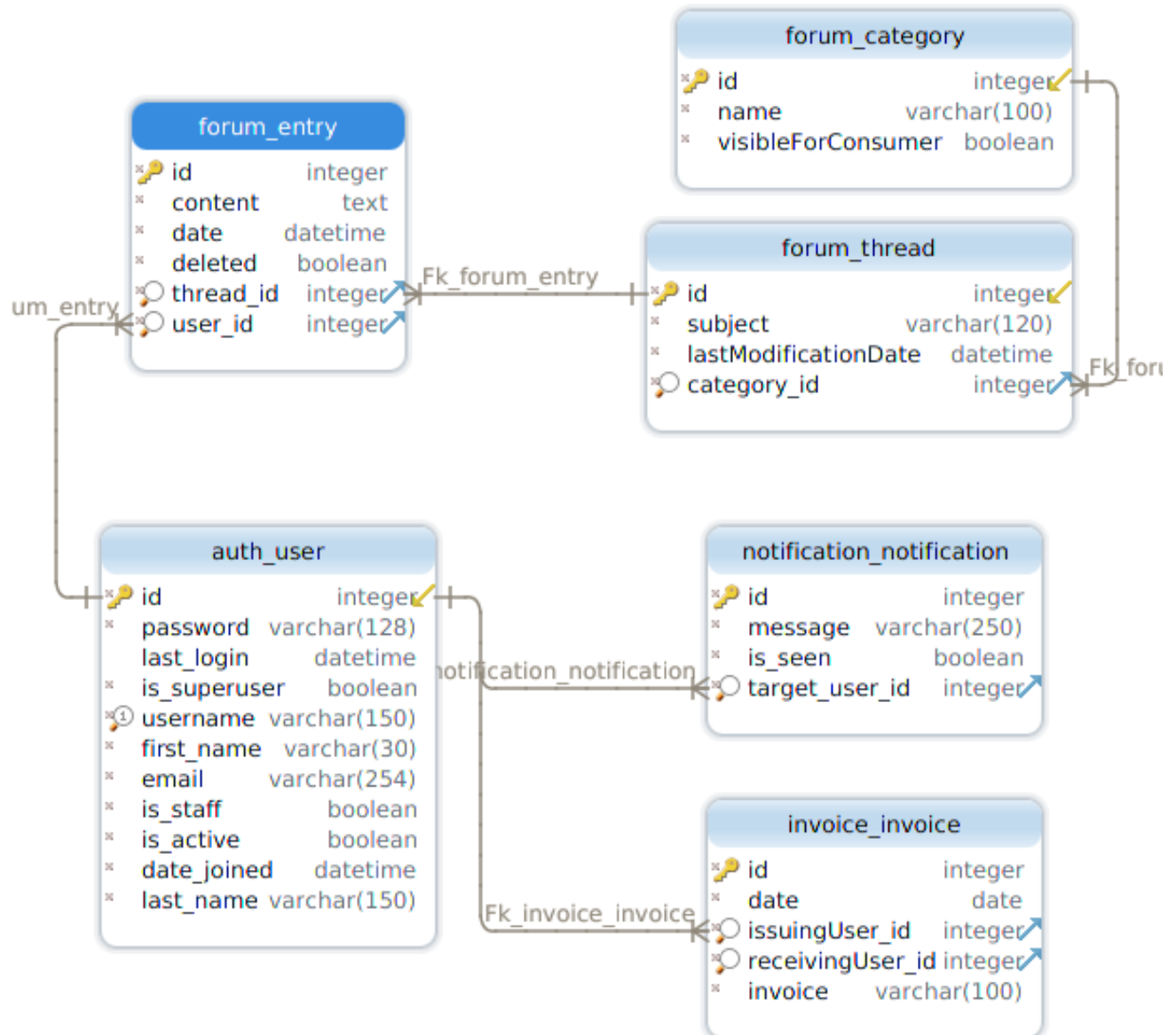
```
{% if user.username != "admin" %}
    <h1>Nothing worth seeing here {{ user.username }}!</h1>
{% else %}
    <h1>Administration view:</h1>
    <table>
        <tr>
            <th>Username</th>
            <th>Firstname</th>
            <th>Lastname</th>
            <th>E-mail</th>
        </tr>
        {% for user in users %}
            <tr>
                <th>{{ user.username }}</th>
                <th>{{ user.first_name }}</th>
                <th>{{ user.last_name }}</th>
                <th>{{ user.email }}</th>
            </tr>
        {% endfor %}
    </table>
{% endif %}
```

Rysunek 25: Zabezpieczenie szablonu przed niepoprawnym wyświetleniem

Oprócz tego wdrożono podstawową autoryzację uniemożliwiającą dostęp do stron bez zalogowania.

6.3 Diagram związków encji

Został on przedstawiony w notacji Martina (kruczej stopki)



Rysunek 26: Diagram związków encji

6.4 Mapowanie ORM

Baza danych jest tworzona na podstawie zdefiniowanych klas, dla przykładu forum:

```
class Category(models.Model):
    name = models.CharField(max_length=100)
    visibleForConsumer = models.BooleanField(default=False)

    def __str__(self):
        return self.name

class Thread(models.Model):
    subject = models.CharField(max_length=120)
    category = models.ForeignKey(Category, on_delete=models.DO_NOTHING)
    lastModificationDate = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.subject

class Entry(models.Model):
    user = models.ForeignKey(User, on_delete=models.DO_NOTHING)
    thread = models.ForeignKey(Thread, on_delete=models.PROTECT)
    content = models.TextField(max_length=2000, help_text="Enter your post here.")
    date = models.DateTimeField(auto_now=True)
    deleted = models.BooleanField(default=False)
```

Rysunek 27: Model ORM forum

6.5 Widoki

Widoki są podstawą jeżeli mowa o komunikacji z bazą danych i użytkownikiem. Udostępniają poszczególne metody, które reagują odpowiednio zależnie od odebranych danych.

```

def post(self, request):
    form = self.form_class(request.POST, request.FILES)
    active_user = request.user

    if active_user is not None:
        if active_user.is_active:
            if form.is_valid():
                invoice = form.save(commit=False)

                invoice.issuingUser = form.cleaned_data['issuingUser']
                invoice.receivingUser = form.cleaned_data['receivingUser']
                invoice.date = datetime.datetime.now()
                invoice.invoice = form.cleaned_data['invoice']

                if invoice.issuingUser is None:
                    return HttpResponse("<h1>UNAUTHORIZED</h1>")
                if invoice.receivingUser is None:
                    return HttpResponse("<h1>UNAUTHORIZED</h1>")

                invoice.save()
                m = "New invoice!"
                Notification.objects.create(target_user=active_user, message=m)

                wIssuer = Invoice.objects.filter(issuingUser=request.user)
                wReceiver = Invoice.objects.filter(receivingUser=request.user)

                result_list = list(chain(wIssuer, wReceiver))

                context = {
                    'invoices': result_list,
                    'form': form
                }

                return render(request=request, template_name=self.template_name, context=context)
            elif settings.DEBUG:
                print(form.errors)
                return HttpResponse("<h1>NOT form.is_valid()</h1>")

    return redirect('/')

```

Rysunek 28: Metoda post w invoice

```

def index(request, id):
    template = loader.get_template('index.html')

    Notification.objects.filter(id=id).update(is_seen=True)

    context = {
        'notif': Notification.objects.filter(target_user=request.user, is_seen=False)
    }

    return HttpResponse(template.render(context, request))

```

Rysunek 29: Realizacja chowania notyfikacji

```

class UserLoginFormView(View):
    form_class = UserLoginForm
    template_name = 'user/user_login_form.html'

    def get(self, request):
        form = self.form_class(None)
        return render(request, self.template_name, {'form': form})

    def post(self, request):
        form = self.form_class(request.POST)

        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']

            print(username + ' ' + password)

            user = authenticate(username=username, password=password)

            if user is not None:
                if user.is_active:
                    login(request, user)
                    return redirect('/')
            else:
                return HttpResponse(str(form.errors))

```

Rysunek 30: Formularz logowania

6.6 Szablony

Decydują o wyglądzie danych przekazanych przez widok, pozwalają na dynamiczne podejmowanie decyzji podczas ich tworzenia.

```

<h1>Welcome on the index page {{ user.username }}!</h1>
<p>We might not be skilled with CCS but we do make solid things. Or so we think.</p>
<div>
  {% if user.is_authenticated %}
    <h2> You logged in successfully!</h2>
    <h3>Notifications : {{ notif | length }}</h3>
    {% for n in notif %}
      <ul>
        <li> {{ n.message }} <a href="/notification/read/{{ n.id }}">Ok</a></li>
      </ul>
    {% endfor %}
    <h2>Places you might want to visit : </h2>

    <br>
    <ul>
      <li>
        <a href="/user/">User</a>
      </li>
      <li>
        <a href="/invoices/">Invoices</a>
      </li>
      <li>
        <a href="/forum/">Forum</a>
      </li>
    </ul>
    <a href="/user/logout/">logout</a>
  {% else %}
    <h2>Do you want to login or register?</h2>
    <a href="/user/login/">Login</a>
    <a href="/user/register/">Register</a>
  {% endif %}
</div>

```

Rysunek 31: Strona główna

```

<html>
<head>
  <title>Forum</title>
</head>
<body>
  <h1>{{ thread.subject }}</h1>

  {% for post in posts %}
    <p><b>{{ post.user }}</b>, {{ post.date }}</b></p>
    <p>{{ post.content }}</p>
    <br/>
  {% endfor %}

  {% if error_message %}
    <p><strong>{{ error_message }}</strong></p>
  {% endif %}
  <form action="/forum/thread/{{ thread.id }}" method="POST">
    {% csrf_token %}
    {% include 'forum/forms/view.html' %}
    <input type="submit" value="Post">
  </form>
</body>

  <a href="/forum">back to category</a>
  <a href="/">home</a>

</html>

```

Rysunek 32: Forum - szablon konkretnego wątku

6.7 Urls

Istnieją także wzorce do rozwiązywania adresów, na których opierają się widoki, oraz szablony:

```

urlpatterns = [
    path('', views.index),
    path(r'admin/', admin.site.urls),
    path("user/", include("user.urls"), name="user"),
    path("invoices/", include("invoice.urls"), name="invoices"),
    path("forum/", include("forum.urls"), name="forum"),
    path("notification/", include("notification.urls"), name="notifications"),
]

```

Rysunek 33: Podstawowe rozwiązywanie url-i

```
urlpatterns = [
    path('', views.index, name="index"),
    path(r"list/", views.List, name="list"),
    path(r"register/", views.UserFormView.as_view(), name="register"),
    path(r"login/", views.UserLoginFormView.as_view(), name="login"),
    path(r"logout/", views.Logout, name="logout")
]
```

Rysunek 34: Dodatkowe wzorce dla logowania

7 Instrukcja wdrożenia

Instrukcja obejmuje jedynie uruchomienie Django, ze względu na mnogość możliwych konfiguracji.

Należy pobrać repozytorium na serwer:

```
git copy https://github.com/TheMesorio/RemoteDataBases.git
```

Wymagany jest python3 oraz pip3. Tworzymy virtualenv:

```
pip3 install virtualenv
```

```
virtualenv venv
```

```
source venv/bin/activate
```

Następnie instalujemy Django:

```
pip3 install django
```

Możemy zweryfikować poprawność instalacji:

```
django-admin --version
```

Przed uruchomieniem aplikacji należy dokonać migracji:

```
python3 manage.py makemigrations user invoice notification forum
```

```
python3 manage.py migrate
```

Testowe uruchomienie serwera jest możliwe bezpośrednio z frameworka:

```
python manage.py runserver 80
```

Właściwa konfiguracja serwera należy już do osoby wdrażającej aplikację na konkretny typ serwera.

Dla bezpieczeństwa konieczna jest zmiana kluczy i wyłączenie trybu debugowania:


```
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET_KEY = 'h^wmb15iyn9binmljuz784fn1#*lbvwy$1#rq3i@78hvhlgdl7'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED_HOSTS = []
```

Rysunek 35: Ważne opcje bezpieczeństwa

8 Podsumowanie

Dzięki dokładnej analizie potrzeb klienta otrzymaliśmy kompletny zestaw wymagań, które powinny być spełnione. Umożliwiło to lepsze zaprojektowanie modularnego systemu, który można w prosty sposób zmieniać, gdy te wymagania ulegną modyfikacji. Następnie zaimplementowaliśmy niezbędne funkcjonalności oraz przeprowadziliśmy wdrożenie aplikacji.

Finalnie, system wspiera firmę w przekazywaniu faktur klientom. Do informowania klienta jest wykorzystywany także system powiadomień. Przekazuje on informacje o nowych postach na forum, które jest ostatnią funkcjonalnością, jaka została wdrożona. Całość ma za zadanie usprawnić komunikację między firmą kupującą nasze rozwiązanie, a jej klientami.

9 Literatura

- Wykład z Internetowych Baz Danych: <http://roman.ptak.staff.iiar.pwr.wroc.pl/>
- Dokumentacja Pythona 3.6: <https://docs.python.org/3.6/>
- Dokumentacja Django 2.1.2: <https://docs.djangoproject.com/en/2.1/contents/>
- Dokumentacja HTML4: <https://www.w3.org/TR/html4/cover.html>