Akademia Górniczo - Hutnicza

Wydział Fizyki i Informatyki Stosowanej

Informatyka stosowana

Terminy

Dokumentacja deweloperska

Monika Smaza
Kamil Kucharski
Dorian Kossowski
Mateusz Libirt
Marcin Miś
Piotr Majkut

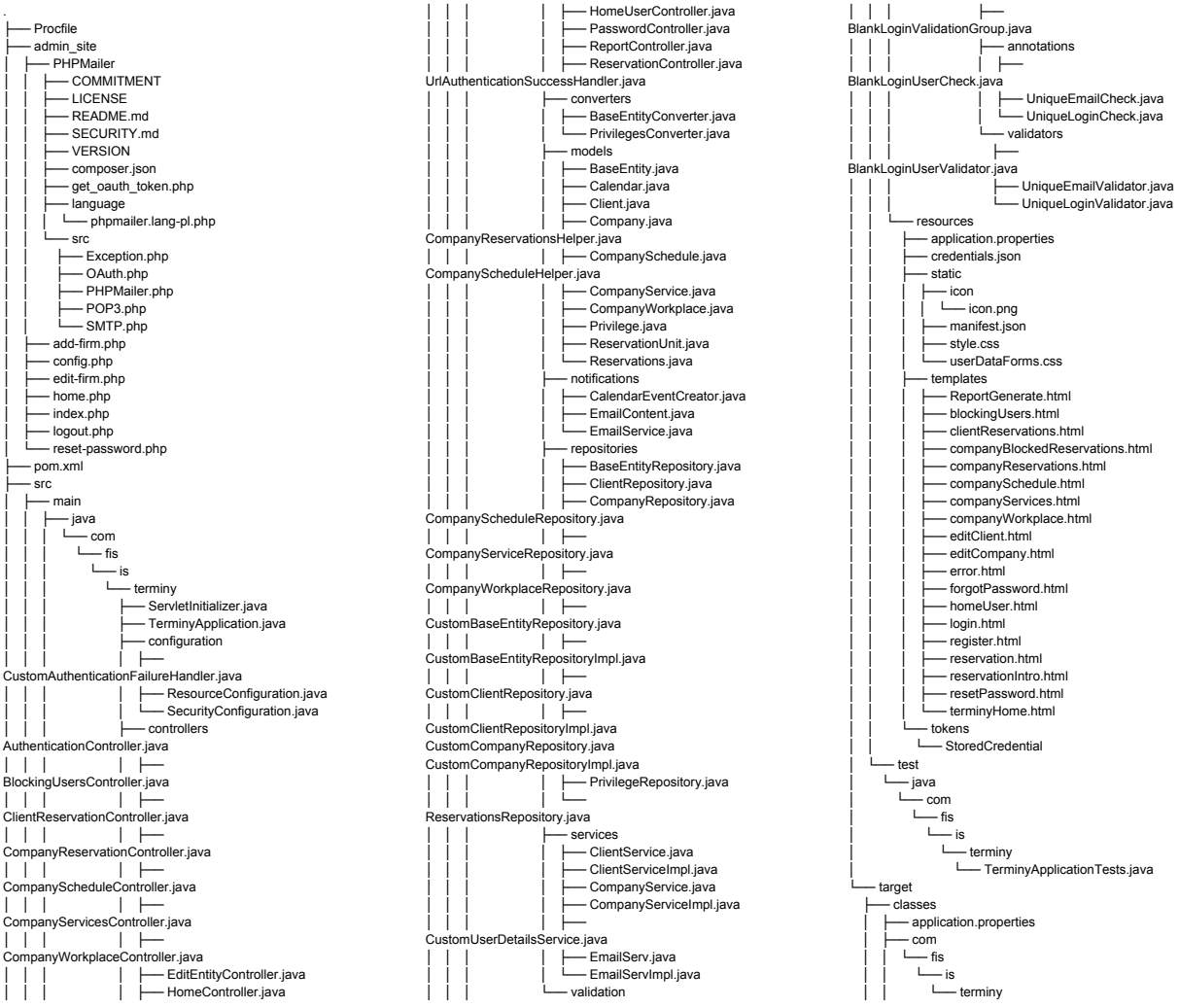
17.06.2019

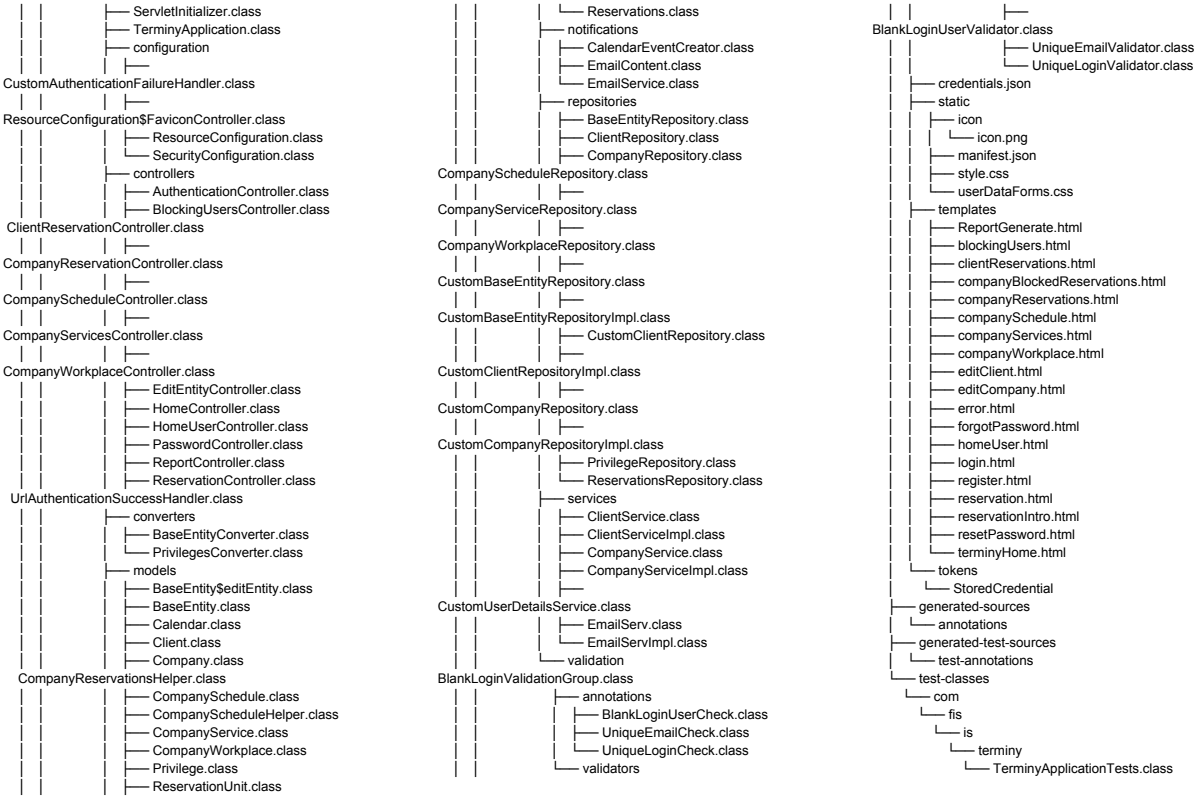
Spis treści

Wykorzystane języki oraz technologie

- Java
- Spring Boot
- Maven
- Hibernate
- PHP
- HTML
- CSS
- Bootstrap

Struktura projektu





Opis Najważniejszych klas

- **TerminyApplication.java**
  Punkt wejścia aplikacji

<u>services</u>:

- **EmailServImpl.java**
  Implementacja interfejsu EmailServ

- **EmailServ.java**
  void sendEmail(SimpleMailMessage email) - funkcja służąca do wysyłania maili

- **CustomUserDetailsService.java**

- **CompanyServiceImpl.java**
  Klasa implementująca interfejs CompanyService

- **CompanyService.java**
  - *Optional<Company> findCompanyByEmail(String email)* - wyszukiwanie potencjalnej firmy za pomocą e-maila
  - *void save(Company company)* - funkcja dodająca nową firmę do bazy

- **ClientServiceImpl.java**
  Klasa implementująca interfejs ClientService

- **ClientService.java**
  - *Optional<Client> findClientByEmail(String email)* - wyszukanie potencjalnego klienta za pomocą e-maila.
  - *void save(Client client)* - zapisanie klienta w bazie

repositories:
Folder zawierający klasy odpowiadające za komunikację z poszczególnymi tabelami bazy danych.

- **ReservationsRepository.java**
  - *findByClientId(Long clientId, Pageable pageable)*
  - *Optional<Reservations> findById(Long id)* - wyszukiwanie rezerwacji za pomocą ID
  - *findByIdAndClientId(Long id, Long clientId)* - wyszukiwanie potencjalnej rezerwacji za pomocą ID firmy oraz ID klienta
  - *findAllByCompanyWorkplaceId(Long companyWorkplaceId)* - wyszukanie listy wszystkich rezerwacji za pomocą ID stanowiska
  - *findAllByCompanyWorkplaceIdAndDate(Long companyWorkplaceId, LocalDate date)* - wyszukanie listy rezerwacji za pomocą ID stanowiska oraz daty
  - *findAllByClientIdAndCompanyWorkplaceId(Long clientId, Long companyWorkplaceId)* - wyszukanie listy rezerwacji za pomocą ID klienta oraz ID stanowiska

- **PrivilegeRepository.java**
  - *findByPrivilege(String privilege)* - wyszukanie przywileju po nazwie

- **CustomCompanyRepositoryImpl.java**
  Klasa implementująca interfejs CustomCompanyRepository
    - *saveModifiedCompany(Company company)* - zapisuje firmę w bazie danych

- **CustomCompanyRepository.java**
  Interfejs dla klasy CustomCompanyRepositoryImpl.java

- **CustomClientRepositoryImpl.java**
    - *saveModifiedClient(Client client)* - zapisuje klienta do bazy danych wraz z określonymi przywilejami

- **CustomClientRepository.java**
  Interfejs dla klasy CustomClientRepositoryImpl.java

- **CustomBaseEntityRepositoryImpl.java**
- **CustomBaseEntityRepository.java**

- **CompanyWorkplaceRepository.java**
    - *findById(Long companyWorkplaceId)* - wyszukanie stanowiska pracy przy pomocy ID firmy
    - *findAllByCompanyId(Long companyId)* - wyszukanie listy stanowisk danej firmy za pomocą jej ID
    - *findByCompanyIdAndName(Long companyId, String name)* - wyszukanie potencjalnego stanowiska za pomocą nazwy stanowiska oraz ID firmy

- **CompanyServiceRepository.java**
    - *Optional<CompanyService> findByIdAndCompanyId(Long id, Long companyId)* - wyszukiwanie usług przy pomocy ID firmy oraz ID usługi
    - *List<CompanyService> findAllByCompanyId(Long companyId)* - wyszukanie wszystkich usług świadczonych przez firmę za pomocą ID firmy

- **CompanyScheduleRepository.java**
    - *findByCompanyWorkplaceId(Long companyWorkplaceId)* - Wyszukiwanie kalendarza firmy za pomocą ID stanowiska
    - *findAllByCompanyWorkplaceId(Long companyWorkplaceId)* - wyszukanie wszystkich stanowisk pracy danej firmy
    - *findById(Long id)* - wyszukanie potencjalnego terminu o podanym id
    - *findByCompanyWorkplaceIdAndDay(Long companyWorkplaceId, String day)* - wyszukanie potencjalnego terminu pracy stanowiska w podanym dniu

- **CompanyRepository.java**
    - *findByLogin(String login)* - odpowiedzialną za wyszukiwanie firmy za pomocą loginu
    - *findByCodedName(String codedName)* - odpowiedzialną za wyszukiwanie firmy za pomocą HashCode
    - *findByMail(String mail)* - odpowiedzialną za wyszukiwanie firmy za pomocą maila

- **ClientRepository.java**
  Interfejs zawierający metody:
    - *findByLogin(String login)* - odpowiedzialną za wyszukiwanie klienta za pomocą loginu
    - *findByMail(String mail)* - odpowiedzialną za wyszukiwanie klienta za pomocą maila
    - *findById(Long id)* - odpowiedzialną za wyszukiwanie klienta za pomocą Id

<u>notifications</u>:
Folder przechowujący klasy służące do powiadamiania firmę/klientów poprzez wiadomości email, a także tworzący wydarzenia do kalendarza google.

- **EmailService.java**
  Klasa służąca do wysyłania maili, korzystająca z biblioteki JavaMailSender.

- **EmailContent.java**
  Klasa będąca odpowiednikiem zawartości maila, zawiera informacje o nadawcy, odbiorcy oraz o treści maila.

- **CalendarEventCreator.java**
  Klasa korzystająca z API udostępnionego przez Google pozwalająca link do wydarzenia, który jest wyświetlany podczas rezerwacji terminu i umożliwia zapisanie go w kalendarzu klienta.

<u>models</u>:
Folder zawierający klasy odpowiadające strukturą tabel bazy danych (jedna klasa to jedna tabela), ponadto zawiera klasy pomocnicze. Każda struktura zawiera metody będące getterami i setterami dla poszczególnych pól znajdujących się w tabeli. Poniżej znajduje się krótki opis klas wraz z metodami nie służącymi do ustawiania/pobierania pól.

- **Reservations.java**
  Klasa mapująca tabele z rezerwacjami. Zawiera pola:
  - *Long id* - id (unikalne id rezerwacji)
  - *CompanyWorkplace companyWorkplace* - Stanowisko na które została zarezerwowana rezerwacja
  - *BaseEntity client* - Klient rezerwujący
  - *CompanyService service* - Usługa
  - *LocalDate date* - Data rezerwacji
  - *LocalTime start_hour* - godzina rozpoczęcia rezerwacji
  - *LocalTime end_hour* - godzina końca rezerwacji

- **ReservationUnit.java**
  Klasa pomocnicza służąca do przygotowywania rezerwacji do zapisania oraz zawierająca najważniejsze pola tabeli z rezerwacjami. Określa konkretną instancje rezerwacji. Zawiera pola:
  - *Long id* - id rezerwacji
  - *LocalTime start_hour* - godzina rozpoczęcia
  - *LocalTime end_hour* - godzina zakończenia

- **Privilege.java**
  Klasa mapująca tabele z przywilejami firmy (wysyłanie maili, możliwość blokowania klientów, generowanie raportów). Zawiera pola:
  - *Long id* - unikalne id przywileju
  - *String privilege* - nazwa przywileju

- **CompanyWorkplace.java**
  Klasa mapująca tabele stanowisk pracy firmy. Zawiera pola:
  - *Long id* - unikalne id stanowiska pracy
  - *String name* - nazwa stanowiska
  - *Company company* - firma udostępniająca dane stanowisko

- **CompanyService.java**
  Klasa mapująca usługi firmy. Zawiera pola:
  - *Long id* - unikalne id usługi
  - *Company company* - firma udostępniająca usługę
  - *String name* - nazwa usługi
  - *Long price* - cena usługi
  - *Long duration* - czas trwania usługi wyrażony w minutach

- **CompanyScheduleHelper.java**
  Klasa pomocnicza przygotowująca czas pracy danego stanowiska dla firmy, a także służąca jako blokowanie rezerwacji danego stanowiska w danym dniu przez firmę. Zawiera pola:
  - *String day* - nazwa dnia pracy
  - *LocalDate date* - data rezerwacji (przydatne w przypadku rezerwacji blokowania rezerwacji użytkownikom przez firmę)
  - *LocalTime start_hour* - godzina rozpoczęcia
  - *LocalTime end_hour* - godzina zakończenia
  - *String workplaceName* - nazwa stanowiska

- **CompanySchedule.java**
  Klasa mapująca tabelę kalendarza firmy (czasu pracy) na danym stanowisku. Zawiera pola:
  - *Long id* - unikalne id terminu
  - *String day* - dzień tygodnia danego terminu
  - *LocalTime start_hour* - godzina początkowa pracy
  - *LocalTime end_hour* - godzina zakończenia pracy
  - *CompanyWorkplace companyWorkplace* - stanowisko firmy odpowiadające danemu kalendarzowi

- **Company.java**
  Klasa mapująca tabela firm. Klasa dziedziczy po BaseEntity. Zawiera pola:
  - *String phone* - numer telefonu firmy
  - *String mail* - adres email firmy
  - *String name* - nazwa firmy
  - *String codedName* - nazwa kodowana firmy
  - *String resetToken* - token używany do generowania jednorazowych maili

- **Client.java**
  Klasa dziedzicząca po BaseEntity mapująca tabele klientów zawierająca pola analogiczne do klasy Company bez pola codedName za to z dodatkowym polem surname oznaczającym nazwisko klienta.

- **Calendar.java**
  Klasa pomocnicza, określająca czas i dzień, a także tłumacząca angielskie nazwy dni na polskie odpowiedniki.

- **BaseEntity.java**
  Absrtakcyjna klasa będąca podstawowym modelem dla klas Client i Company. Mapująca odpowiadająca jej tabele w bazie danych. Zawiera pola:
  - *Long id* - unikalne id
  - *String login* - login firmy/użytkownika
  - *String password* - zakodowane hasło firmy/użytkownika

<u>controllers</u>:
Folder zawierający główną część aplikacji. Poszczególne klasy i ich metody służą do mapowania adresów URL przesyłanych metodami GET/POST/DELETE. Każda klasa korzysta z odpowiednich modeli i repozytoriów w celu kierowania zapytań do bazy danych.

- **ReservationController.java**
  Klasa odpowiadająca za wyświetlanie wolnych terminów oraz rezerwowanie. Funkcjonalność po stronie klienta.

- **ReportController.java**
  Klasa odpowiadająca za dostarczenie odpowiednich danych w celu generowania raportów w formie wykresów.

- **PasswordController.java**
  Klasa odpowiedzialna za działanie w przypadku zapomnienia hasła. Użytkownik ma możliwość stworzenia nowego hasła - w tym przypadku na jego mail podany podczas rejestracji jest wysyłany jednorazowy link z możliwością wpisania nowego hasła.

- **EditEntityController.java**
  Klasa odpowiedzialna za zmianę danych firmy/użytkownika takich jak: mail, numer telefonu, nazwę oraz hasło.

- **CompanyWorkplaceController.java**
  Klasa odpowiadająca za tworzenie/usuwanie stanowisk pracy wraz z odpowiednią nazwą po stronie firmy.

- **CompanyServicesController.java**
  Klasa odpowiedzialna za dodawanie usług wraz z czasem trwania i ceną wykonania. Umożliwia również usuwanie konkretnych usług. Funkcja działa po stronie firmy.

- **CompanyScheduleController.java**
  Klasa odpowiedzialna za dodawanie czasu pracy w konkretnym dniu dla poszczególnego stanowiska. Funkcja działa po stronie firmy.

- **CompanyReservationController.java**
  Klasa odpowiedzialna za wszystkie rezerwacje firmy. Umożliwia wypisanie wszystkich zarezerwowanych terminów, a także odwołanie konkretnego terminu wraz z informacją mailową do klienta.

- **ClientReservationController.java**
  Klasa odpowiedzialna za wszystkie rezerwacje po stronie klienta. Umożliwia wypisanie wszystkich zarezerwowanych terminów przez użytkownika dla danej firmy, a także rezerwację konkretnej usługi w wyznaczonym dniu.

- **BlockingUsersController.java**
  Klasa odpowiedzialna za blokowanie użytkowników. Jeżeli dany użytkownik zarezerwował termin firma ma możliwość zablokowania takiego użytkownika. Kontroler odpowiada za wyświetlanie wszystkich użytkowników z możliwością zablokowania ich.

- **AuthenticationController.java**
  Klasa odpowiadająca za poprawną walidacje podczas logowania.

Pomysły na ulepszenie serwisu

- Dodanie możliwości wysyłania powiadomień za pomocą SMS (wykorzystanie biblioteki SMSAPI dostępnej w językach: Java, C#, python, PHP)
- Dodanie mechanizmu logowania aktywności aplikacji np. logowanie informacji o zablokowaniu użytkownika (użycie jednej z dostępnych w javie bibliotek do logowania np. Java Logging API, Log4J, LogBack, TiniLock)
- Usprawnienie wyświetlania możliwych terminów rezerwacji (np. w postaci tabeli dla kilku stanowisk równocześnie)
- Możliwość filtrowania wszystkich wyświetlanych rezerwacji po konkretnych stanowiskach czy usługach. Wykonać to można dodając dodatkowy warunek do zapytania do bazy danych lub filtrując już bezpośrednio w wyższych warstwach.