**ТЕМА:**

**“Система за управление на медицински изследвания.”**

**(Medical examination management system)**

**ПРЕДМЕТ:**

**„Разработка на уеб приложения и услуги**

**със Spring Framework v5“**

**Изготвили:**

Aнтония Тодорова

Павлина Колева

София Михалева

София

12.02.2020

**СЪДЪРЖАНИЕ**

ОПИСАНИЕ НА ОСНОВНИТЕ ФУНЦИОНАЛНИ И НЕФУНКЦИОНАЛНИ ИЗИСКВАНИЯ - - - - - - - - - - - - - - - - - - - - - ( 3 )

ПОТРЕБИТЕЛСКИ СЛУЧАИ - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 4 )

ИЗПОЛЗВАНИ ТЕХНОЛОГИИ И БИБЛИОТЕКИ - - - - - - - - - - - - ( 5 )

АРХИТЕКТУРА НА СИСТЕМАТА И ВЗАИМОДЕЙСТВИЕ

МЕЖДУ ОБЕКТИТЕ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 6 )

ИНТЕРФЕЙСИ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 7 )

ОПИСАНИЕ НА НАЧИНА НА ИНСТАЛИРАНЕ И

КОНФИГУРИРАНЕ НА РАЗРАБОТЕНАТА СИСТЕМА - - - - - - - - - ( 11 )

РАБОТА СЪС СИСТЕМАТА - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 12 )

ЗАКЛЮЧЕНИЕ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 20 )

ИЗТОЧНИЦИ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 20 )

**1. Описание на основните функционални и нефункционални изисквания**

Разработената система за управление на медицински изследвания предоставя **удобен начин за следене и управление на главните процеси в медицинска лаборатория за изследвания**. Web приложението улеснява както пациентите, така и лекарите и ръководителите на медицинското заведение.

Благодарение на системата екипът на лабораторията разполага с **лесен и интуитивен интерфейс** за:

* **следене на записаните часове**, както в цялото медицинско заведение, така и по медицински отделения
* **промяна на статуса на завършеност на изследване**
* **проследяване на завършеността на проведените изследвания**
* **добавяне на резултат от извършено изследване**
* **преглед на медицинския картон на пациент**

Чрез бързия и подреден достъп до тези информация медицинският екип разполага с възможността за **по-добро планиране и организиране на работния процес**.

Други съществени плюсове от използването на системата са:

* **спестяването на време**, защото цялата необходима информация е събрана на едно място.
* **предотвратяване на застъпващи се или дублиращи се часове** за конкретен кабинет.
* осигуряването на **допълнителна сигурност и конфиденциалност на информацията** чрез съществуването на различни потребителски роли. По този начин система гарантира, че всеки неин потребител ще има достъп само до предназначените за него данни.

Системата е в полза и за пациентите, които получават възможност след регистрацията си в системата:

* да проверяват свободните часове за конкретно отделение
* да си запазят час за изследване
* да проверят статуса на скорошните си изследвания
* да прегледат резултатите си от всички проведени в лабораторията изследвания
* да видят медицинския си картон

**2. Потребителски случаи**

* Регистриране - всеки нов потребител се регистрира в системата, въвеждайки потребителско име, имена, егн, парола и валиден имейл, като системата валидира полетата и при невалидно такова изписва съответния проблем. Стандартната страница за регистрация създава само потребители с роля “PATIENT”. Създаването на потребители с други роли, трябва да се извърши от лице с админски прави, което гарантира сигурността в системата.  
  Админът има възможност да създава доктори, пациенти или други админи.
* Потребителски профили - Всеки пациент може да види личната си информация в профила си - имена и егн. Администраторът има правата да управлява текущите потребители - да блокира акаунтите им и да ги отключва. Всеки администратор може да търси потребители в списъка по тяхното потребителско име.
* Създаване на ново клинично отделение - всеки администратор има правата да създава ново клинично отделение, задавайки му име и стая, в която се намира.
* История на резултати от медицински изследвания - всеки пациент може да разглежда свои стари изследвания в историята на медицинския си картон. За всяко изследване той може да намери подробна информация за резулта и датата на направеното изследване.
* Всеки доктор има достъп до всички резултати на пациентите от своето отделение и до тяхното подробно представяне.

Всеки администратор има достъп до всички потребители и до техните медицински картони. Той може да разгледа изследванията на всеки един регистриран потребител.

* Управляване на часове - Регистриран пациент може да вижда какви клинични отделения има и да избере едно от тях, за което да си запази час в интервала от 7 дни в седмицата от 06:00 часа до 22:00 часа през един час. Това се случва чрез избиране на дата и време за провеждане на изследването.

Всеки пациент може да вижда всичките си запазени часове.

Всеки доктор може да вижда графика си - какви часове са запазени за неговото отделение.

Администраторът може да разглежда всички запазени часове в системата.

* Статус на изследване - пациентът може да вижда статуса на своите изследвания. Той може да проследи кога изследването е започнато, в процес, обработващо се или е затворено, което значи че резултатът е готов.

Докторът може да променя статуса на изследванията на пациентите в своите клинични отделения.

Администраторът може да променя статуса на изследванията във всички клинични отделения.

* Резултати - Докторът може да добавя резултат към изследванията, със статус pending, в отделението, в което работи. Наличните в момента резултати са от следните два вида: кръвни или ядрено магнитни. За кръвните резултати доктор въвежда стойности за всеки един от параметрите и описание, ако има такова, а за ядрено магнитните - добавя снимка от изследването и описание, ако е необходимо.

Aдминистраторът също може да добавя резултати за всички клинични отделения по описания по-горе начин.

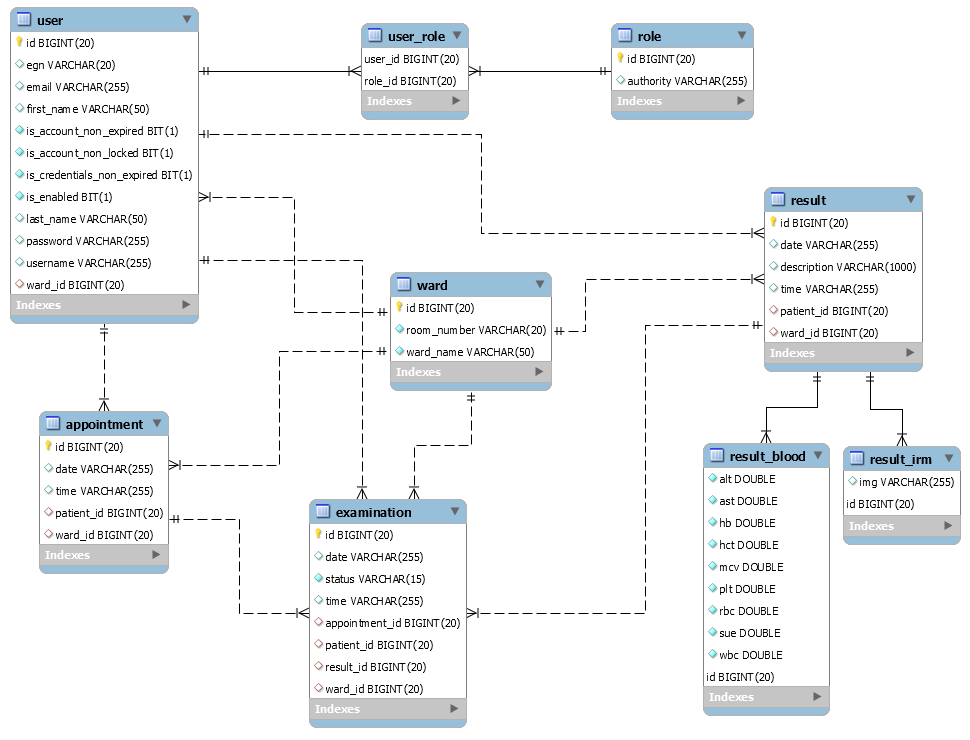
* Статистика - администраторът може да вижда статистика за броя на пациентите, броя на всички резултати и на различните видове резултати, направени чрез системата.

**3.** **Използвани технологии и библиотеки**

За реализацията на backend-a на системата е използван Spring 5 Application Development Framework и Spring Boot. Тези мощни технологии позволяват лесната конфигурация на проекта. Front-end изгледите са реализирани чрез Thymeleaf темплейти. За по-добра визуализация е използвана библиотеката Bootstrap.  
Рутирането между страниците се извършва server side чрез SpringMVC. Базата, използвана за проекта е MySQL. Проектът използва Liquibase - open source библиотека за лесни манипулации и конфигурации на базата.

**4.** **Архитектура на системата и взаимодействие между обектите**

Следната диаграма описва архитектурата на база данни, върху която системата работи, и илюстрира взаимоотношенията между основните обекти:



Взето е архитектурното решение в системата да се поддържат различни потребителски роли:

* ADMIN
* PATIENT
* DOCTOR

За момента всеки регистриран потребител е асоцииран точно с една потребителска роля, но е предвидено разширяването на проекта и е възможно добавянето на нови междинни потребителски роли или по-финно разграничаване на потребителските права чрез добавяне на ”granted authorities”, които описват множеството от разрешени операции за всеки потребител. Например в момента за потребители с роля “PATIENT” e позволено само преглеждането на собствените си изследвания и резултати. А на потребители с роля “DOCTOR” е разрешен само достъпът до запазените часове, изследвания и резултати към отделението, в което работят.

Взаимодействието между потребител и медицинско отделение се определя от ролята на потребителя. Ако потребителят е асоцииран с роля на пациент или админ, той не е свързан към никое медицинско отделение, но за потребители с роля “DOCTOR” е въведено ограничение да принадлежат на точно едно медицинско отделение и то да е това, в което работят. Следователно, когато потребител с роля “ADMIN” създава акаунт на лекар е необходимо админът да уточни отделението, в което практикува новосъздаденият лекар.

Всяко изследване в системата се характеризира чрез статус на завършеност. Възможните статуси, в които може да се намира изследване са следните:

* open - изследвания, които предстоят да бъдат извършени
* processed - изследвания, които се извършват в момента
* pending - изследвания, чиито резултат се обработва в момента
* closed - завършени изследвания, чиито резултат е готов

Друго съществено уточнение, което трябва да се направи, засяга особеността, че при запазване на час за изследване автоматично се създава изследване със статус “open”, съответстващо на записания час, както и при промяна на статуса на изследване от “pending” на “closed” се налага въвеждане на резултата от изследването.

**5.** **Интерфейси**

Web приложението освен графичен интерфейс предоставя и REST API, което да направи възможно и лесно свързването и взаимодействието на системата със други third party софтуерни решения. REST API-ито включва основните CRUD операции за всеки top-level domain aggregate, а именно:

* Roles API:
  + GET /api/roles - връща списък от всички потребителски роли в системата
  + GET /api/roles/{id} - връща конкретна потребителска роля по id, или съобщение за грешка, ако роля с това id не съществува.
  + POST /api/roles - създава нова потребителска роля. Трябва да се характеризира с “authority” (id автоматично се попълва от MEMS и създадения обект се връща като резултат от POST заявката).
  + PUT /api/roles/{id} - промяна на потребителска роля, промененият ресурс се връща като отговор на заявката. Ако id в URL и това в подадения JSON обект не съвпадат се връща грешка.
  + DELETE /api/roles/{id} - изтриване на потребителска роля, изтритият ресурс се връща като отговор на заявката
* Wards API:
  + GET /api/wards - връща списък от всички медицински отделения в системата
  + GET /api/wards/{id} - връща конкретно медицинско отделение по id, или съобщение за грешка, ако медицинско отделение с това id не съществува.
  + POST /api/wards - създава ново медицинско отделение.Трябва да се характеризира с “wardName” и “roomNumber” (id автоматично се попълва от MEMS и създадения обект се връща като резултат от POST заявката).
  + PUT /api/wards/{id} - промяна на медицинско отделение, промененият ресурс се връща като отговор на заявката. Ако id в URL и това в подадения JSON обект не съвпадат се връща грешка.
  + DELETE /api/wards/{id} - изтриване на медицинско отделение, изтритият ресурс се връща като отговор на заявката
* Users API:
  + GET /api/users - връща списък от всички потребители в системата. За потребителите с роли “PATIENT” или “ADMIN” полето “ward” е null, а за потребители с роля “DOCTOR” полето “ward” съдържа информация за отделението, в което работи съответният доктор.
  + GET /api/users/{id} - връща конкретен потребител по id, или съобщение за грешка, ако потребител с това id не съществува.
  + POST /api/users - създава нов потребител.Трябва да се характеризира с “username”, “password”, “email”, “firstName”, “lastName”, “egn”, “authorities” и “wardName” само и единствено, ако потребителят има роля на доктор. В случай че не съществува отделение с такова “wardName” се връща грешка. (id автоматично се попълва от MEMS и създадения обект се връща като резултат от POST заявката).
  + PUT /api/users/{id} - промяна на потребител, промененият ресурс се връща като отговор на заявката. Ако id в URL и това в подадения JSON обект не съвпадат се връща грешка.
  + DELETE /api/users/{id} - изтриване на потребител, изтритият ресурс се връща като отговор на заявката
* Appointments API:
  + GET /api/appointments - връща списък от всички запазени часове за изследвания в системата.
  + GET /api/appointments/patients/{patientId} - връща списък от всички запазени часове за изследвания на конкретен пациент.
  + GET /api/appointments/wards/{wardId} - връща списък от всички запазени часове за изследвания за конкретно медицинско отделение.
  + GET /api/appointments/{id} - връща конкретен запазен час по id, или съобщение за грешка, ако запазен час с това id не съществува.
  + POST /api/appointments - създава нов запазен час. Трябва да се характеризира с “egn” на пациента, който ще се изследва, “wardName” на отделението, в което ще се извършва изследването, “date” и “time”. Прави се проверка дали този час не е зает и ако вече съществува записан час в това време за съответното отделение се връща съобщение за грешка. В случай, че не съществуват пациент с такова “egn” или отделение с такова “wardName” се връща грешка. (id автоматично се попълва от MEMS и създадения обект се връща като резултат от POST заявката).
  + PUT /api/appointments/{id} - промяна на запазен час, промененият ресурс се връща като отговор на заявката. Ако id в URL и това в подадения JSON обект не съвпадат се връща грешка.
  + DELETE /api/appointments/{id} - изтриване на запазен час, изтритият ресурс се връща като отговор на заявката
* Examinations API:
  + GET /api/examinations - връща списък от всички изследвания в системата.
  + GET /api/examinations/statuses/{status} - връща списък от всички изследвания в системата с определен статус. Поддържаните статуси са open, processed, pending и closed.
  + GET /api/examinations/patients/{patientId} - връща списък от всички изследвания на конкретен пациент.
  + GET /api/examinations/patients/{patientId}/statuses/{status} - връща списък от всички изследвания с определен статус на конкретен пациент.
  + GET /api/examinations/wards/{wardId} - връща списък от всички изследвания за конкретно медицинско отделение.
  + GET /api/examinations/wards/{wardId}/statuses/{status} - връща списък от всички изследвания с определен статус за конкретно медицинско отделение.
  + GET /api/examinations/{id} - връща конкретно изследване по id, или съобщение за грешка, ако изследване с това id не съществува.
  + POST /api/examinations - създава ново изследване. Трябва да се характеризира с “egn” на пациента, за който се отнася, “wardName” на отделението, в което се извършва изследването, “date” и “time”. Възможно е уточняването на “appointmentId”, ако изследването е в следствие от предварително записан час, в такъв случай се изисква предварителното създаване на часът чрез POST заявка към /api/appointments. В случай, че не съществуват пациент с такова “egn”, отделение с такова “wardName” или запазен час с такова id се връща грешка. (id автоматично се попълва от MEMS и създадения обект се връща като резултат от POST заявката).
  + PUT /api/examinations/{id} - промяна на изследване, промененият ресурс се връща като отговор на заявката. Ако статусът на изследването се промени на “closed” следва предварително да е създаден резултат, съответстващ на това изследване чрез POST заявка към /api/results/blood или /api/results/irm и неговото id да бъде попълнено в полето с името “resultId”. Ако id в URL и това в подадения JSON обект не съвпадат се връща грешка.
  + DELETE /api/examinations/{id} - изтриване на изследване, изтритият ресурс се връща като отговор на заявката.
* Results API:
  + GET /api/results - връща списък от всички резултати от изследвания в системата.
  + GET /api/results/patients/{patientId} - връща списък от всички резултати от изследвания на конкретен пациент.
  + GET /api/results/wards/{wardId} - връща списък от всички резултати от изследвания за конкретно медицинско отделение.
  + GET /api/results/{id} - връща конкретен резултат от изследване по id, или съобщение за грешка, ако резултат с това id не съществува.
  + POST /api/results/irm - създава нов резултат от ядрено-магнитно изследване. Резултатът да се характеризира с “egn” на пациента, “wardName” на отделението, в което е извършено изследването, “date”, “time”, “description” и “img”. В случай, че не съществуват пациент с такова “egn” или отделение с такова “wardName” се връща грешка. (id автоматично се попълва от MEMS и създадения обект се връща като резултат от POST заявката).
  + POST /api/results/blood - създава нов резултат от кръвно изследване. Резултатът да се характеризира с “egn” на пациента, “wardName” на отделението, в което е извършено изследването, “date”, “time”, “description” и кръвните показатели. В случай, че не съществуват пациент с такова “egn” или отделение с такова “wardName” се връща грешка. (id автоматично се попълва от MEMS и създадения обект се връща като резултат от POST заявката).
  + PUT /api/results/irm/{id} - промяна на ядрено-магнитен резултат, промененият ресурс се връща като отговор на заявката. Ако id в URL и това в подадения JSON обект не съвпадат се връща грешка.
  + PUT /api/results/irm/{id} - промяна на кръвен резултат, промененият ресурс се връща като отговор на заявката. Ако id в URL и това в подадения JSON обект не съвпадат се връща грешка.
  + DELETE /api/results/{id} - изтриване на резултат, изтритият ресурс се връща като отговор на заявката.

**6.** **Описание на начина на инсталиране и конфигуриране на разработената система.**

За стартирането на платформата за управление на медицински изследвания е достатъчно само изтеглянето на source кода и компилирането на проекта в IntelliJ IDEA и модификация на главните настройки на проекта като парола и име на базата, те могат да бъдат променени във файла application.properties. Не е необходимо допълнително инсталиране и конфигуриране.

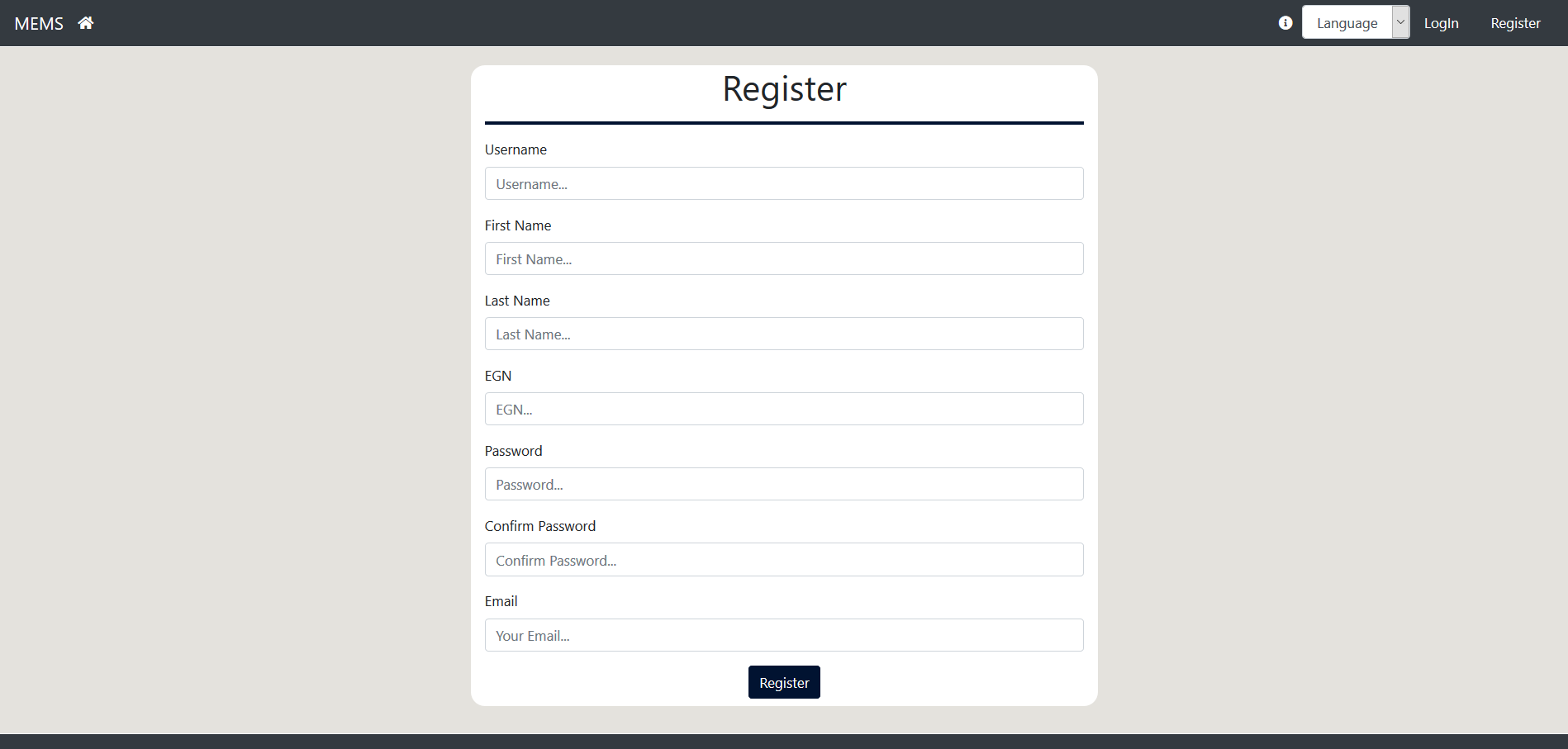
Системата сама създава базата локално, като първоначално се създават трите потребителски роли - админ, доктор и пациент и един потребител, който е с роля админ - с потребителско име и парола admin.

**7.** **Работа със системата**

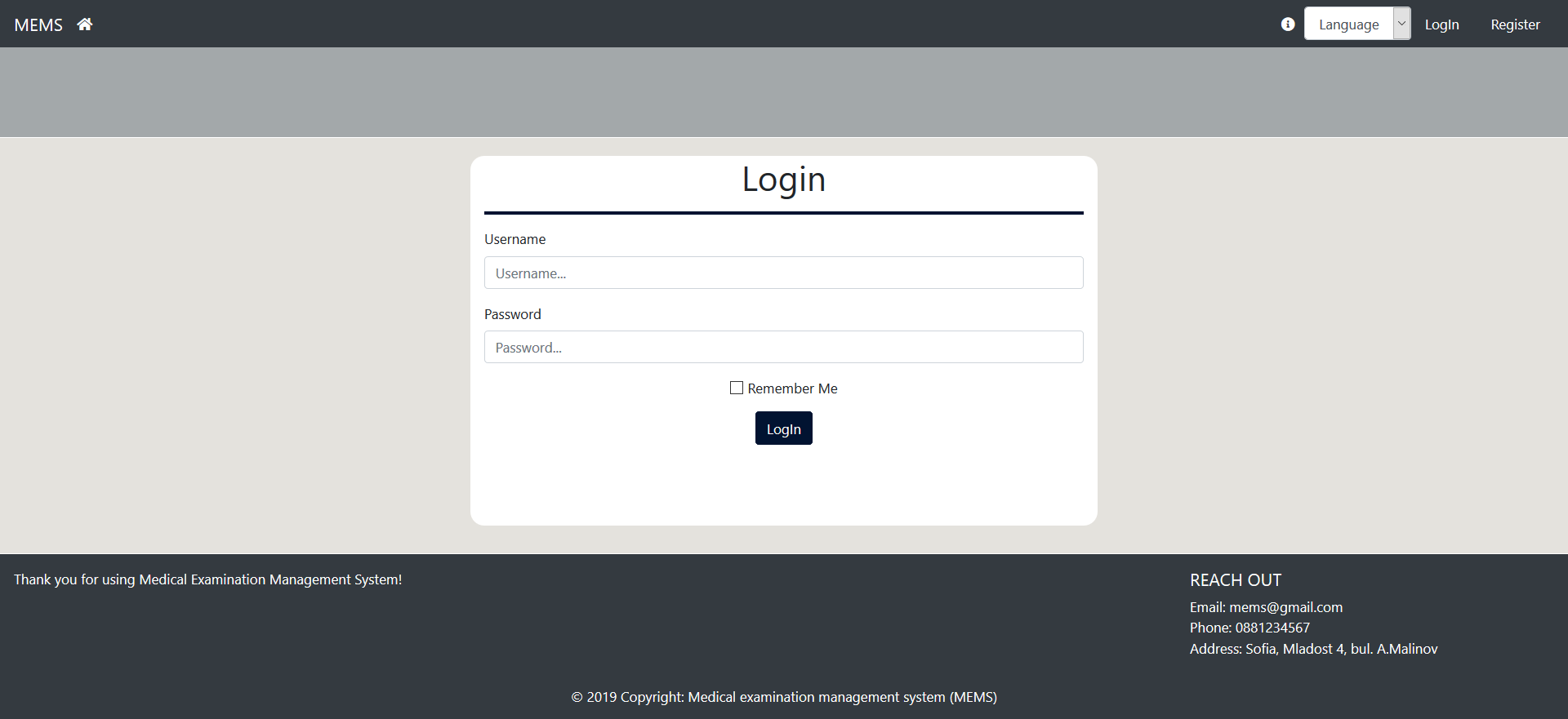
При влизане в системата това е първата страница, която вижда потребителя:



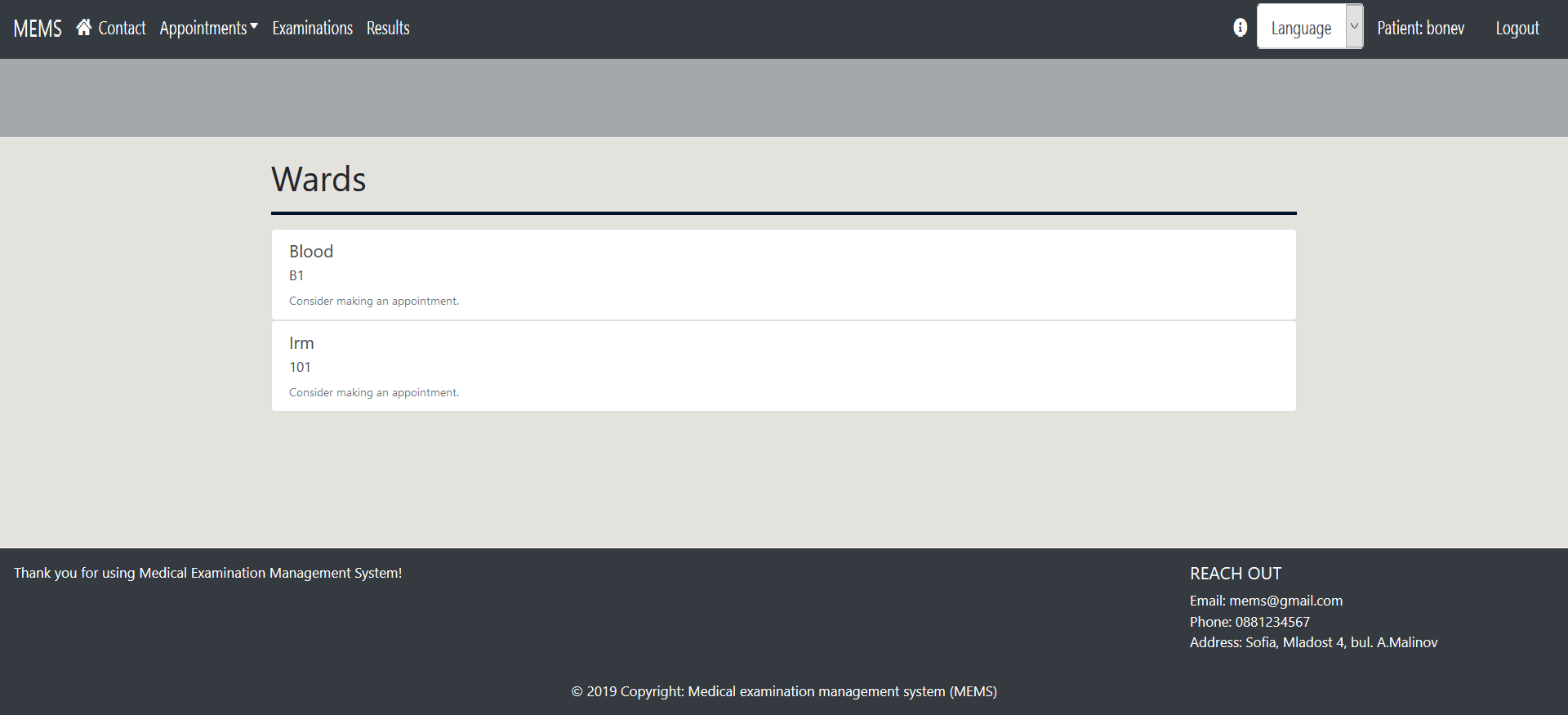
Регистрацията в системата е възможна чрез натискане на бутона Register, намиращ се горе най-вдясно в главното меню на сайта, и попълване на следната форма:



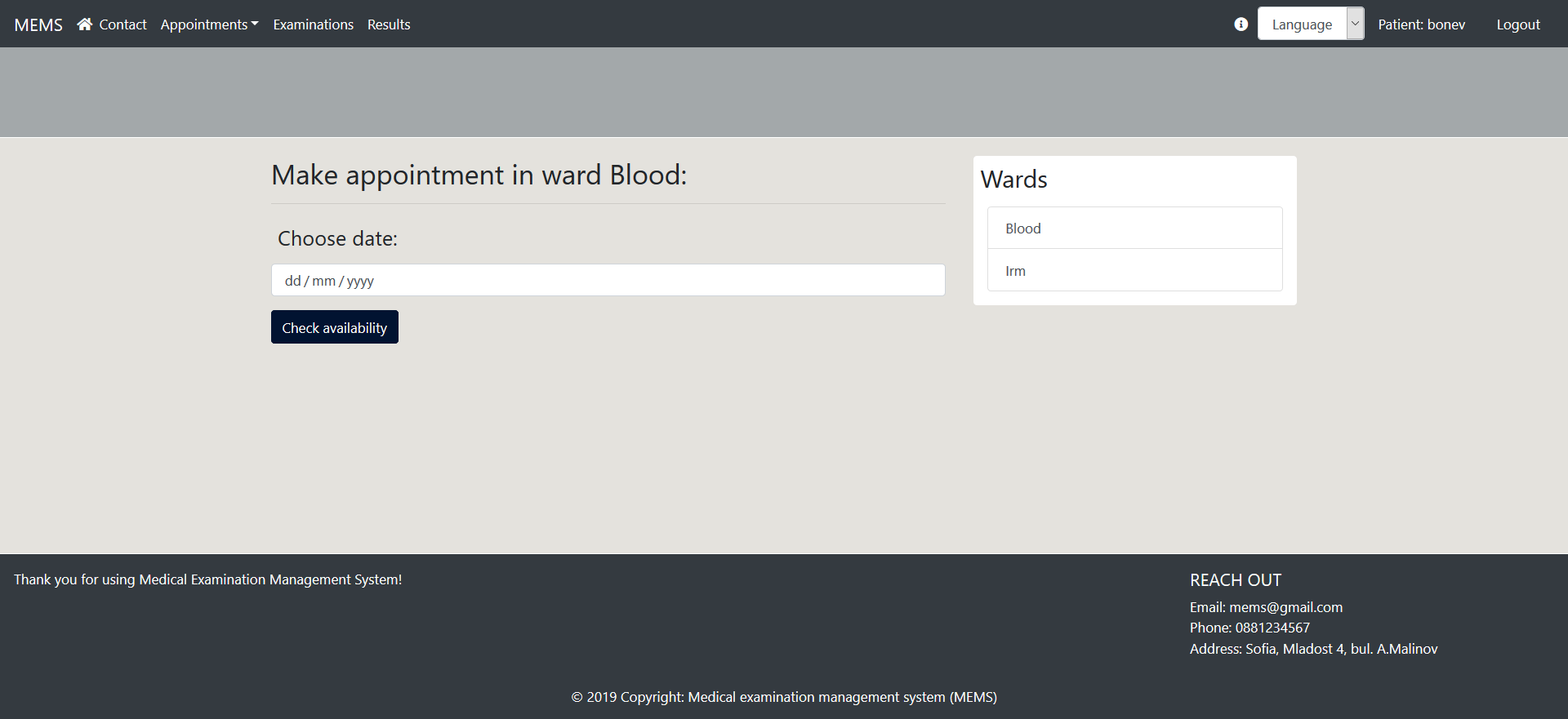
За да влезете в системата като регистриран потребител след това натиснете бутона Login от главното меню и въведете валидно потребителско име и парола в следната страница:



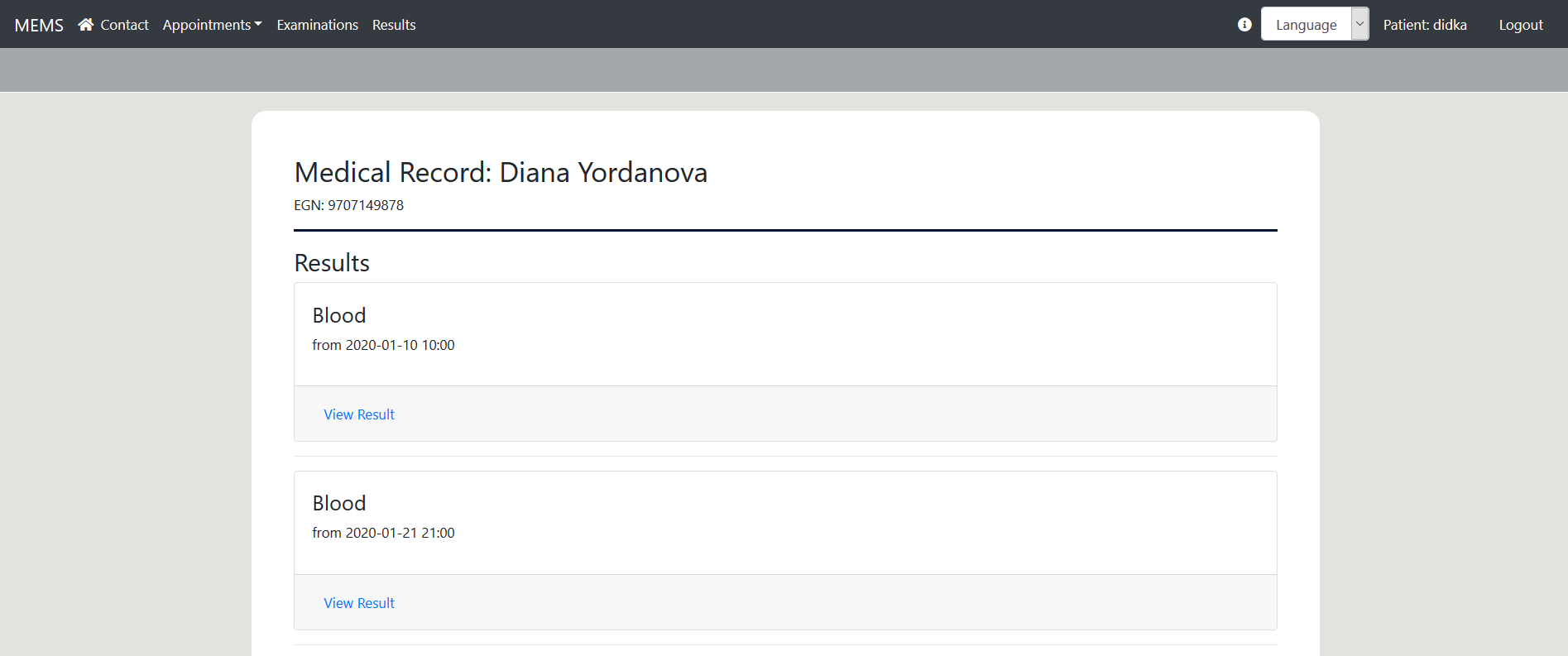
След успешен вход в системата, за да запазите час за изследване отидете на бутона “Appointments” в навигационното меню и изберете опцията “Make Appointment” от dropdown менюто. Ще видите следната страница, показваща всички медицински отделения:



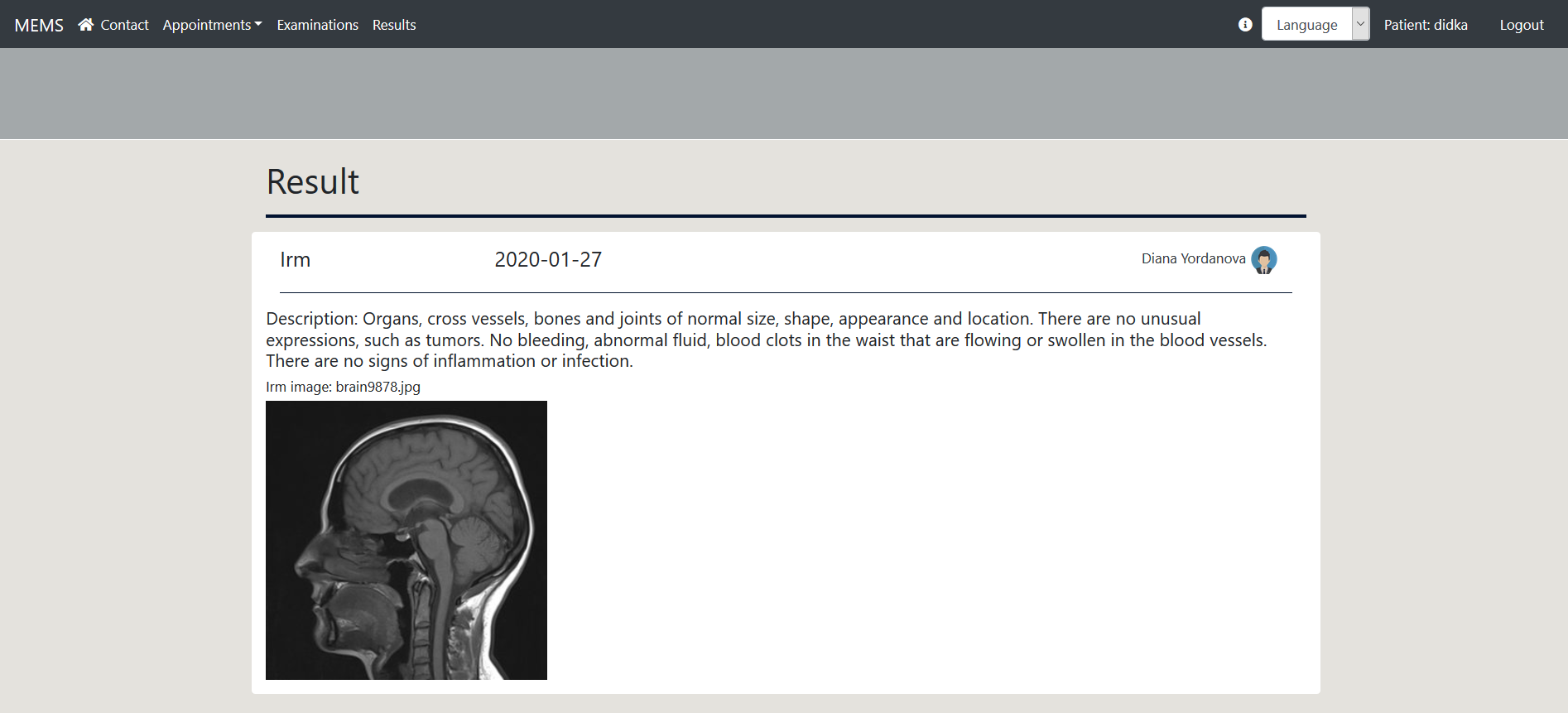
Изберете отделението, което извършва, необходимото ви изследване и изберете ден, в който предпочитате да направите изследването, след това ще имате възможност да видите свободните часове за конкретния ден и да запазите удобен за вас час:



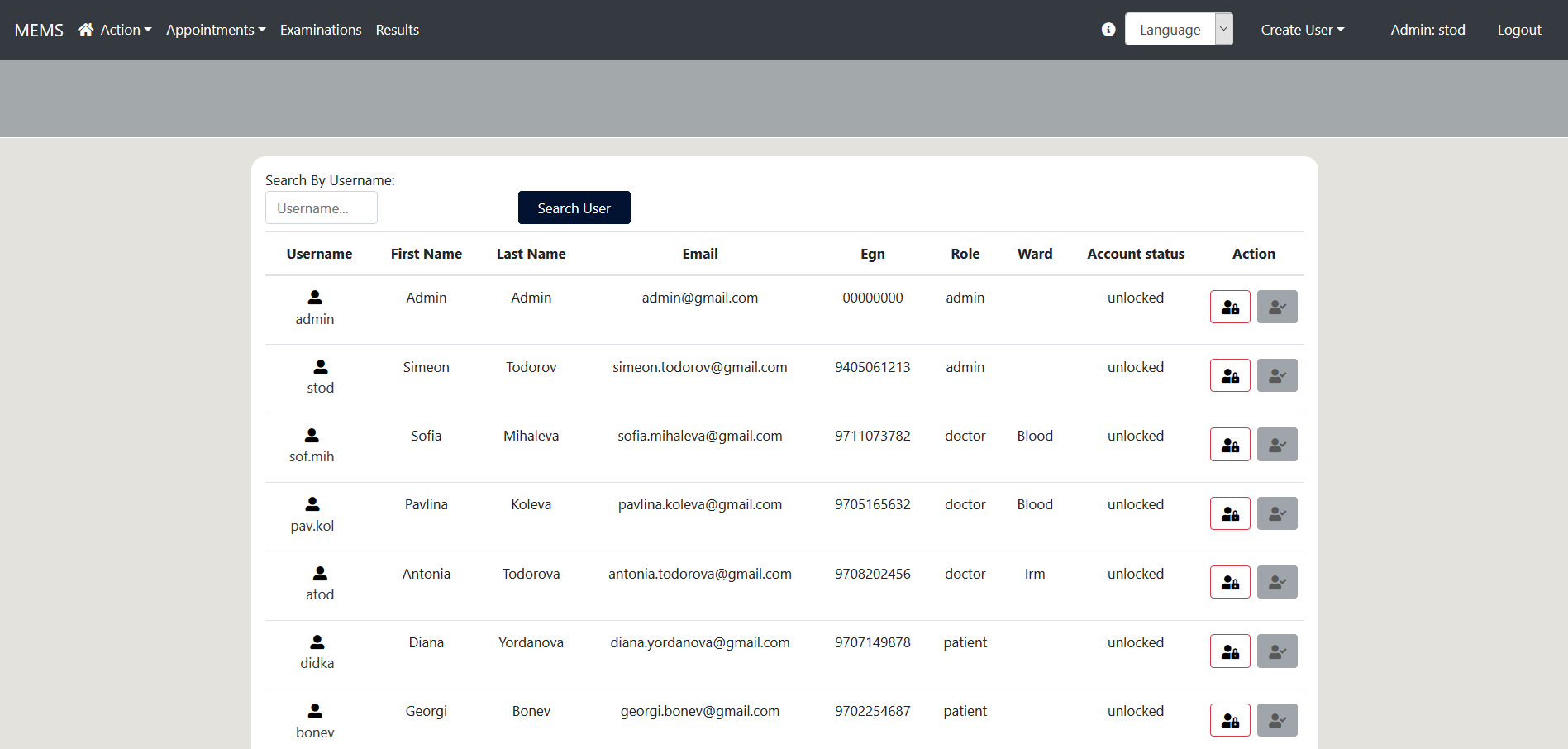
Ако желаете да прегледате медицинския си картон, който съдържа всички ваши изследвания, натиснете вторият (отдясно наляво) бутон в навигационното меню, върху който е изписано Patient:<вашето потребителско име>:



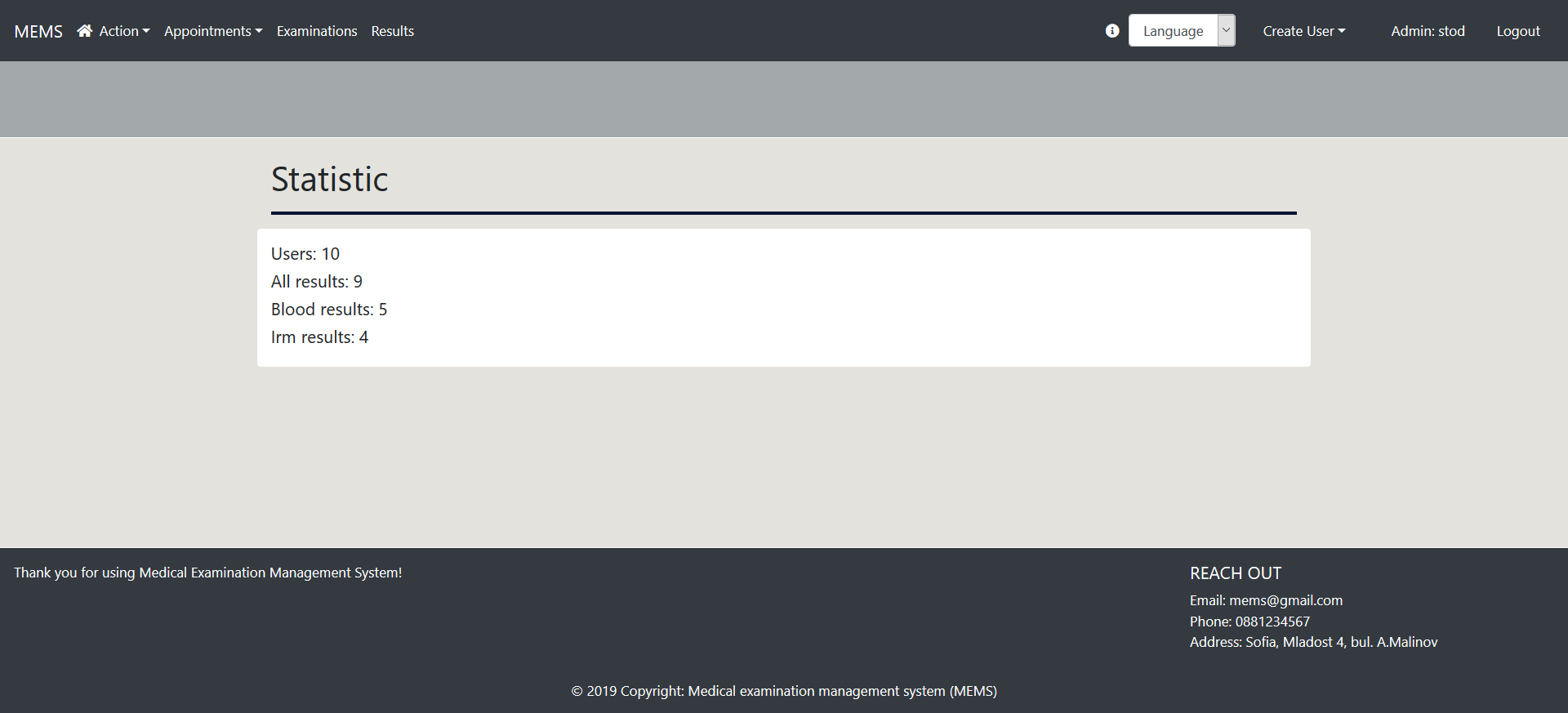
За да видите конкретен резултат, който ви интересува натиснете бутона View Result и в браузъра ви ще се отвори страница, която показва всички детайли за избрания резултат:



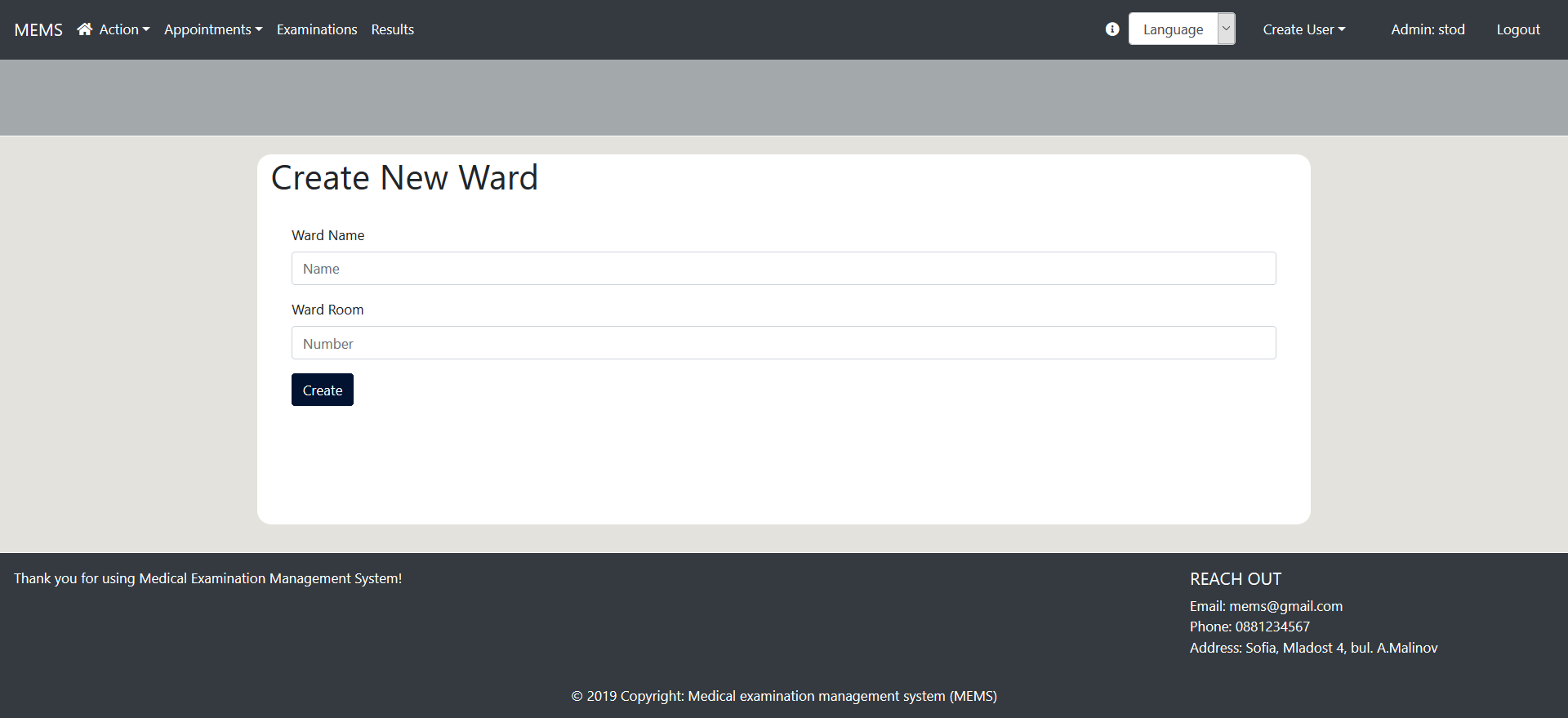
Ако потребителят влезе в системата като админ той може да управлява профилите на останалите потребители при натискане на Action и после избор на действие Manage users:



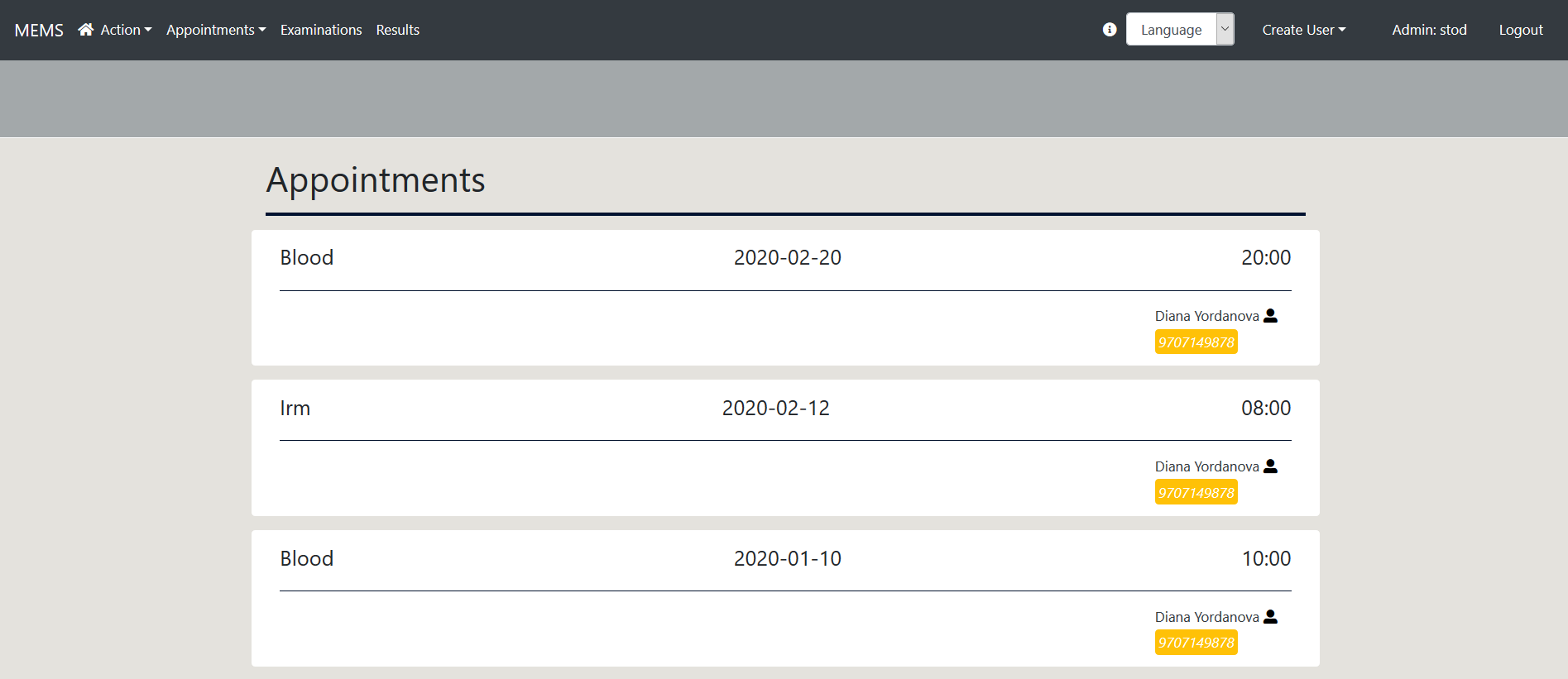
В падащото поле Аction при натискане на опция статистика се визуализира следната информация:



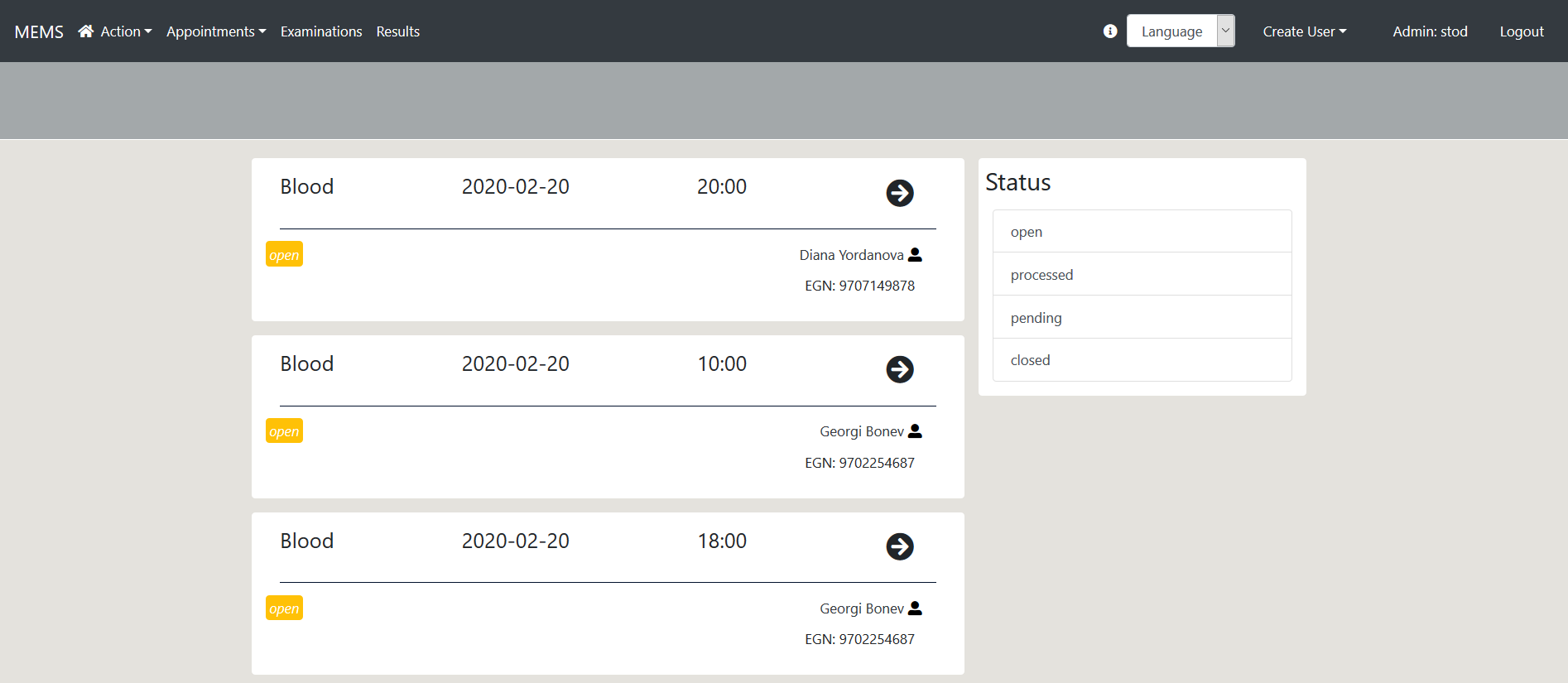
В същото поле Action админът може да създаде отделение чрез избиране на Create Ward и попълване на следната форма:



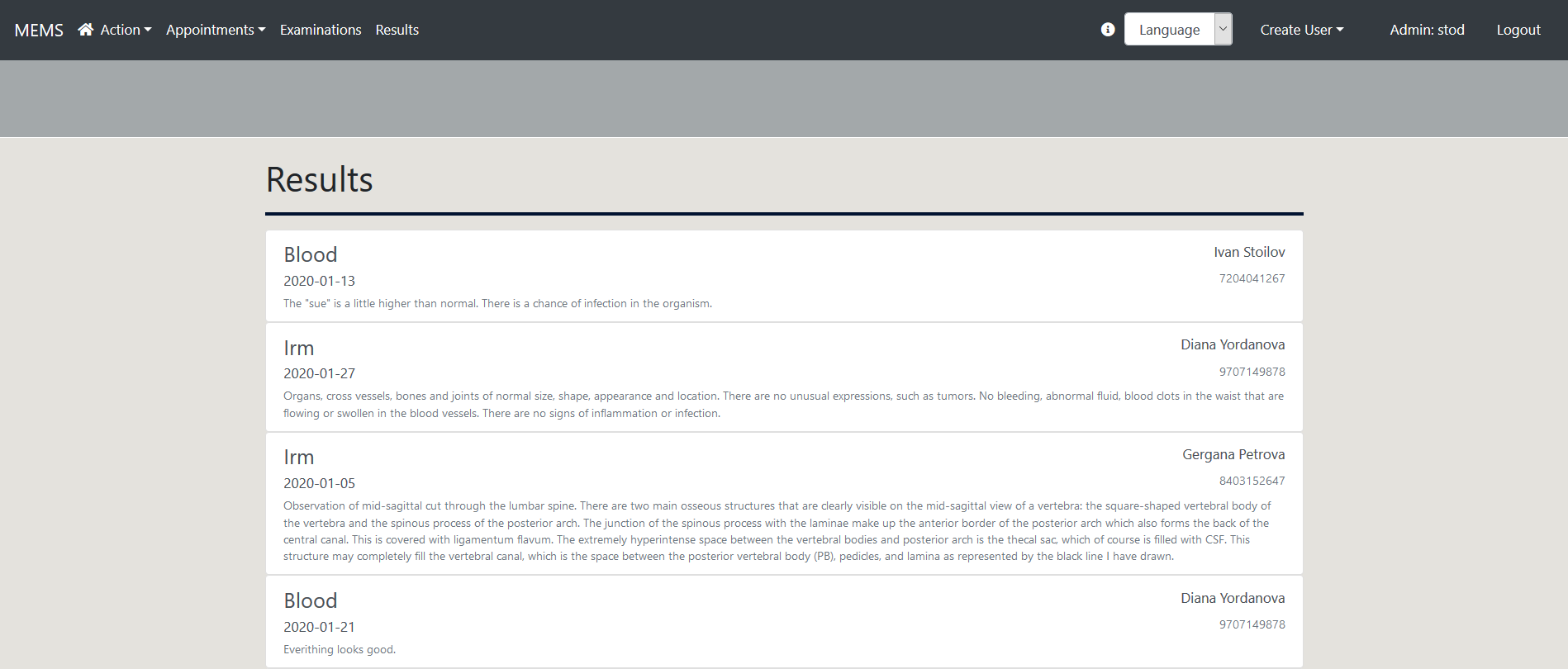
При избиране на Appointments и после View Appointment всички потребители виждат запазените часове, спрямо ролята си (показаните по-долу са за админ):

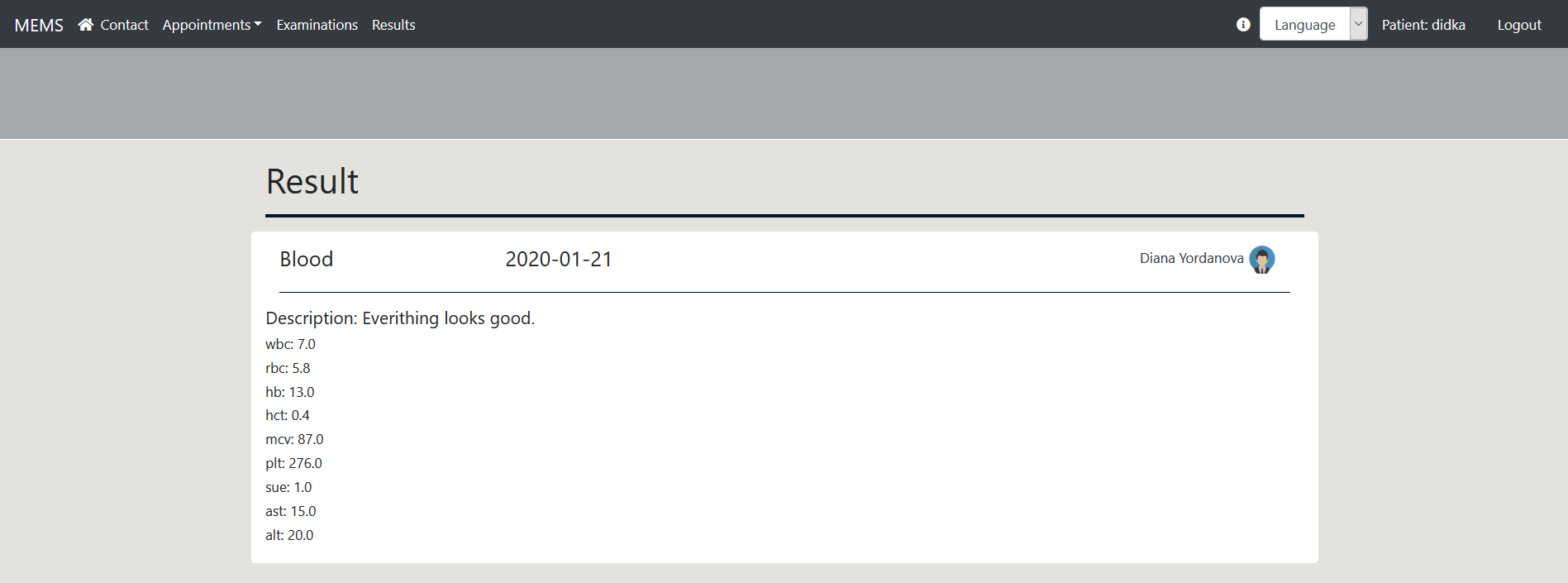


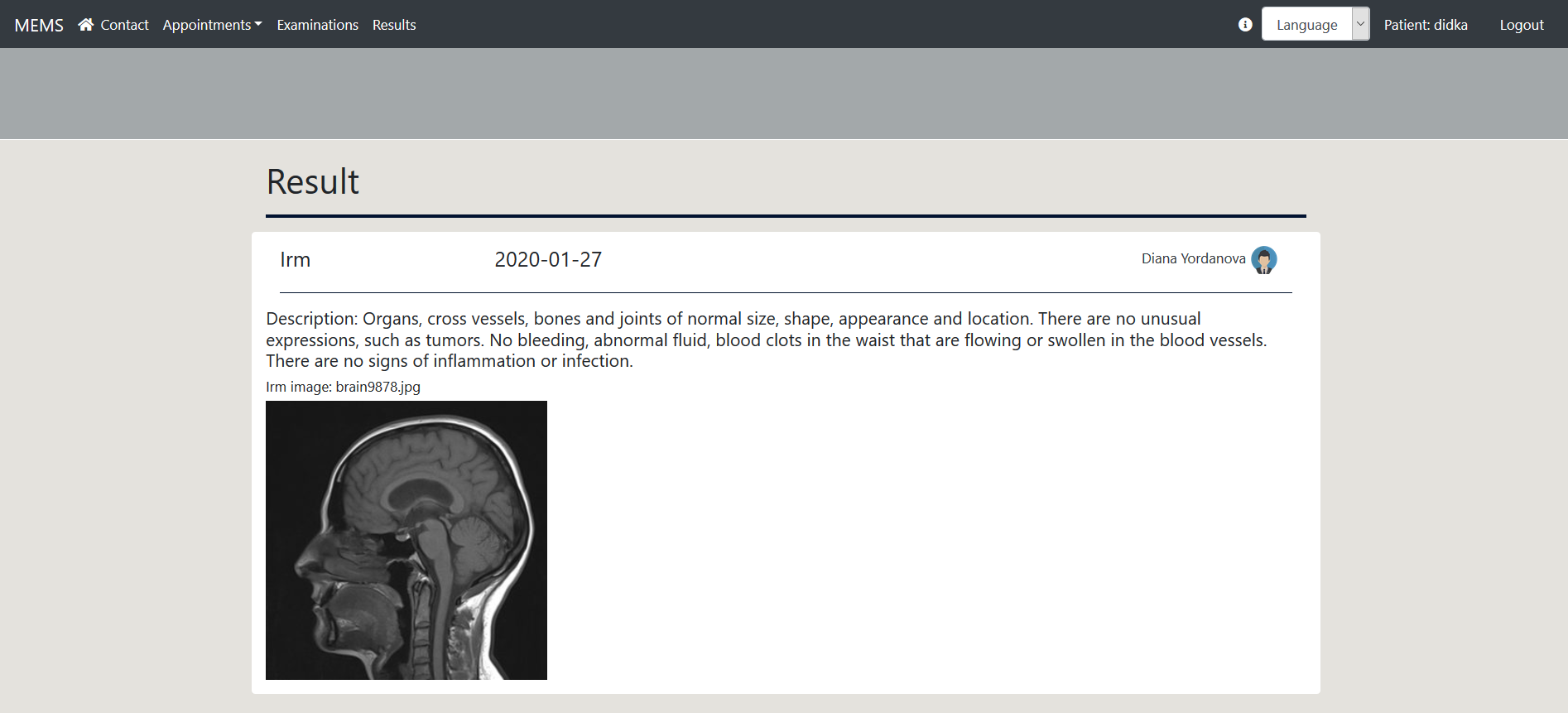
Всеки потребител вижда различна информация в страницата Examinations според ролята си. Чрез секцията Status от дясно на самите изследвания можем да ги филтрираме по статус:



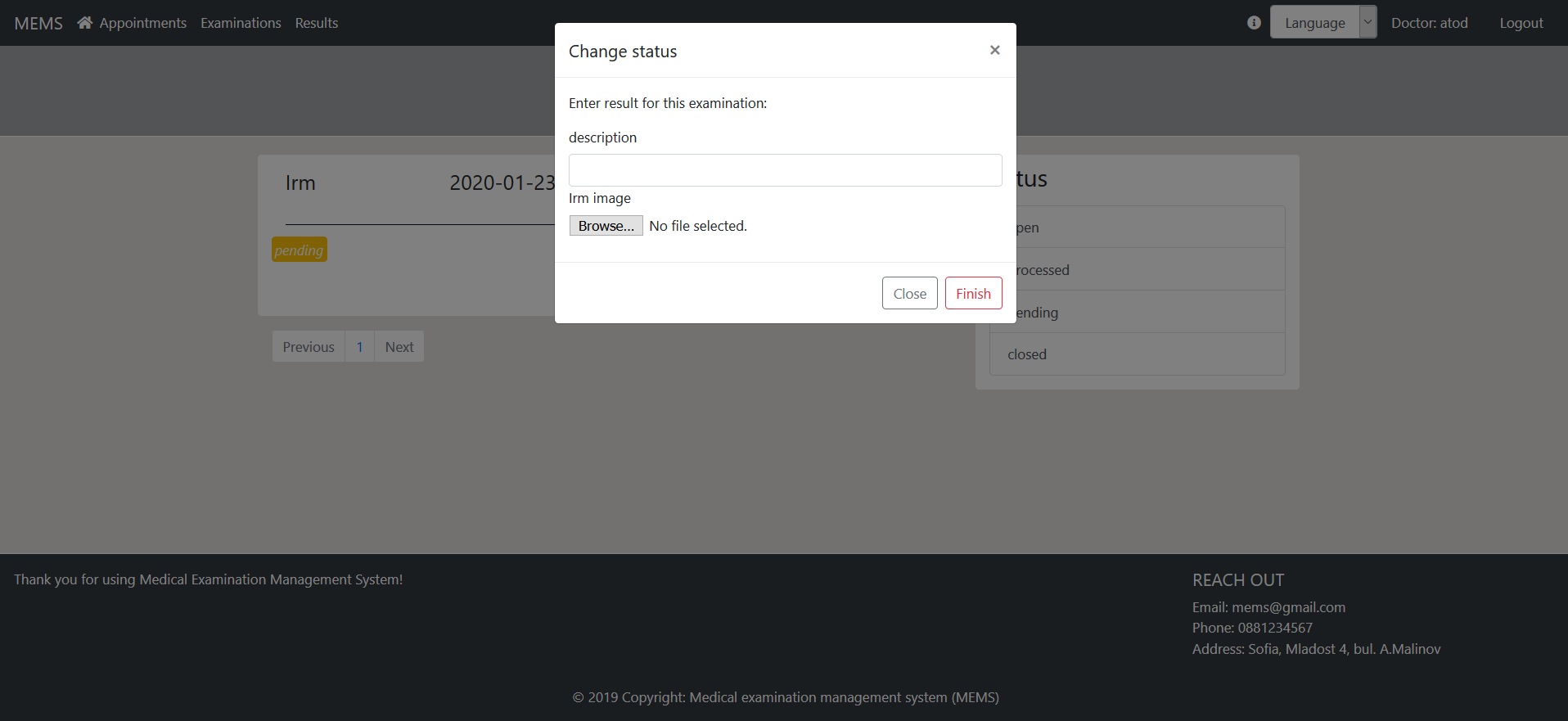
При кликване на Results всеки потребител може да вижда резултатите според ролята си и след това подробна информация за всеки един от тях:

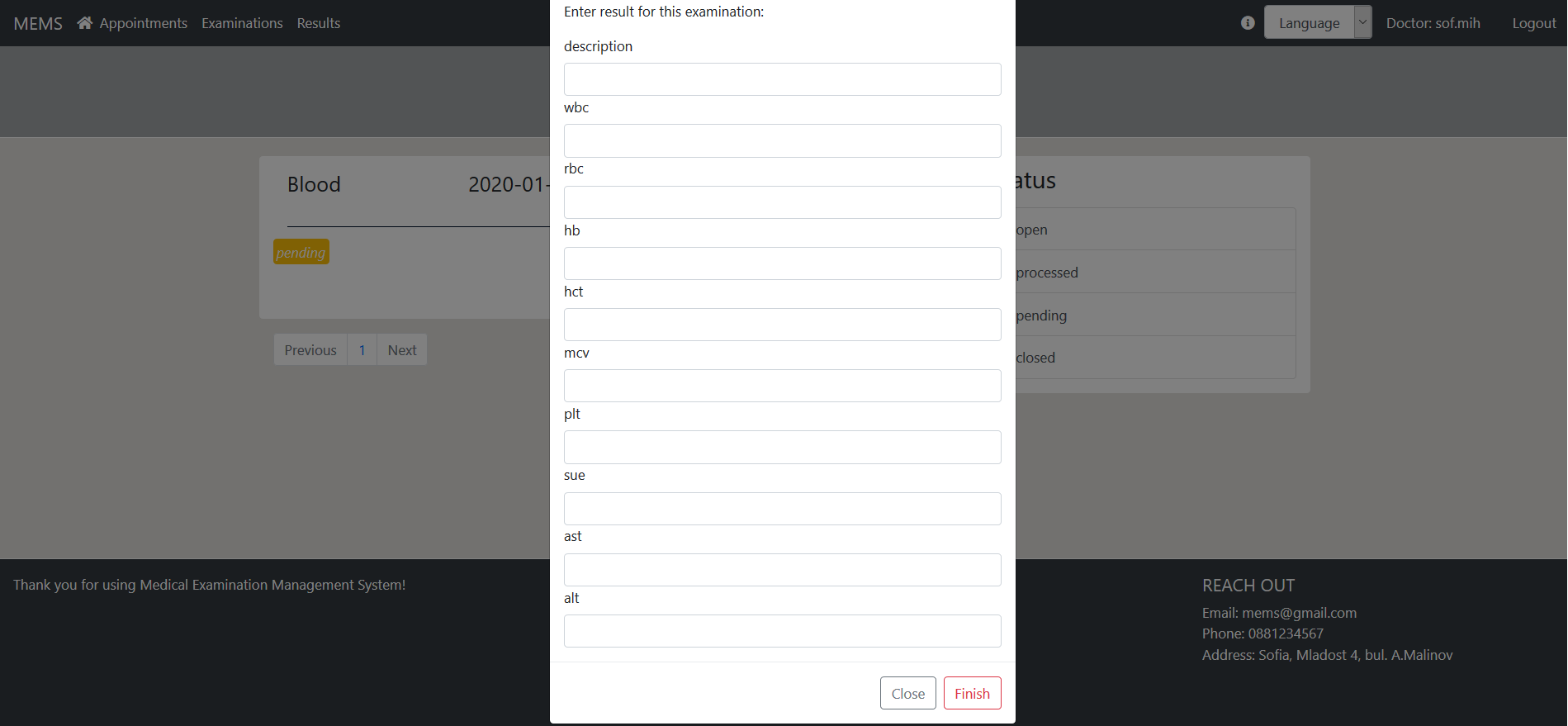






Докторите и админите могат да добавят резултати, което се случва при смяната на статуса на изследванията от pending към closed чрез следните форми:





**8.** **Заключение**

Най-голямото предизвикателство в проекта ни се стори първоначално изграждане на идеята за системата и архитектурата на базата данни. Двоумяхме се за избор на стратегия за наследяване на класовете резултати, както и при реализациите на взаимодействия между обекти от тип One-to-one. Изграждането на front-end частта на проекта беше по-непозната за нас, поради липсата ни на предишен опит, но считаме, че се справихме добре с осъществяването на красив и лесен за употреба front-end.

Възможни подобрения и разширения на наличната функционалност включват:

* възможност за редактиране на персоналната информация в профила на потребител
* възможност за добавяне на нови типове резултати от изследвания

Реализирането на системата беше добра възможност за нас да обогатим знанията си за Spring 5 Application Development Framework.

**9.** **Източници**

Проектът е реализиран чрез множество справки в следните източници:

* <https://github.com/iproduct/course-spring5/wiki>
* <https://thoughts-on-java.org/complete-guide-inheritance-strategies-jpa-hibernate/>
* <https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html>
* <https://www.baeldung.com/hibernate-inheritance>
* <https://www.thymeleaf.org/documentation.html>
* <https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>