

Филиал Московского Государственного Университета
имени М.В. Ломоносова в городе Ташкенте

Факультет прикладной математики и информатики

Кафедра прикладной математики и информатики

Абдуллаева Евгения Гасановна

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**на тему: «Разработка модулей оповещения и статистики
для системы дистанционного обучения»**

по направлению 01.03.02 «Прикладная математика и информатика»

Выпускная квалификационная работа рассмотрена и рекомендована к
защите

Руководитель Филиала МГУ имени М.В. Ломоносова в г. Ташкенте
к.ф.-м.н., доцент Строгалов Александр Сергеевич

Научный руководитель

к.ф.-м.н., в.н.с. Алисейчик Павел Александрович



«____» _____ 2021 г.

Ташкент 2021 г.

Аннотация

В настоящей работе приведены результаты, достигнутые в процессе расширения функциональности системы дистанционного обучения «МГУ Контест»: при разработке модуля оповещения пользователей и модуля статистики, а также при усовершенствовании раздела помощи пользователям.

В модуле оповещения реализованы возможность подписки пользователя на определенные сущности, оповещение пользователей о событиях на сайте, интерфейс настройки оповещений; переработано отображение списка оповещений. В модуле статистики усовершенствована страница профиля студента, добавлено отображение сведений об активности и успеваемости отдельного студента, а также рейтинг студентов; в список курсов внедрено отображение сведений о сложности курсов. Реализована основная страница раздела помощи пользователям.

Abstract

This paper presents the results achieved in the process of expanding the functionality of the “MSU Contest” distance learning system: developing a user notification module and a statistics module, as well as improving the user help section.

The notification module implements the ability to subscribe a user to certain entities, notify users about events on the site, an interface for configuring notifications; redesigned display of the list of notifications. In the statistics module, the student profile page has been improved, the display of information about the activity and progress of an individual student, as well as the rating of students has been added; the display of information about the complexity of courses has been implemented in the list of courses. The main page of the user help section has been implemented.

Содержание

1	Введение	4
2	Общая структура и организация работы	5
3	Модуль оповещения	6
3.1	Подписка на оповещения	6
3.2	Оповещения о событиях на сайте	6
3.3	Список оповещений	7
3.3.1	Бесконечная прокрутка списка оповещений	7
3.3.2	Динамическая отметка оповещений как прочтенных	8
3.4	Интерфейс настройки оповещений	8
3.4.1	Выбор типов получаемых оповещений	8
3.4.2	Список подписок на курсы и разделы	9
4	Модуль статистики	10
4.1	Профиль студента	10
4.1.1	Персонализированные изображения профиля	10
4.1.2	Сведения об активности	10
4.1.3	Результаты курсов	11
4.1.4	Подсчет общей успеваемости и точности студента .	11
4.2	Рейтинг студентов	12
4.3	Сведения о сложности курсов	13
5	Раздел помощи	14
6	Заключение	15
7	Список использованных источников и литературы	16
8	Приложения	17

1 Введение

Участие обучающих интеллектуальных систем в образовательном процессе, в том числе дистанционном, приобретает сегодня все большую значимость. В связи с чем созданная для сопровождения курса «Практикум на ЭВМ» в филиале МГУ им. М.В. Ломоносова в Ташкенте система «МГУ Контест» [3] требует расширения функциональности. При расширении возможностей системы, а следовательно и количества событий, происходящих в системе, возникает необходимость обеспечить для пользователя доступность выбора части оповещений о данных событиях, которые пользователь желает получать. Для полноты информации, получаемой преподавателем о процессе обучения студентов, преподавателям требуется доступ к данным об активности и степени усвоения материала каждым студентом, а также возможность сравнения результатов различных студентов. Эта информация необходима как при разработке индивидуальных программ обучения для студентов, являющейся одной из сформулированных в работе [1] целей создания информационно-познавательной среды, так и при анализе общих результатов преподаваемых курсов. Целью данной выпускной квалификационной работы является решение поставленных проблем с последующим внедрением полученного решения в функционирующую систему «МГУ Контест».

Реализация данных усовершенствований способствует повышению эффективности взаимодействия студентов и преподавателей в системе «МГУ Контест», что позволит сделать шаг к обеспечению максимальной результативности всего цикла дистанционного обучения, описанного в работе [2].

2 Общая структура и организация работы

Данная работа содержит две основные части, описанные в главах 3 и 4. В главе 3 представлено описание работы, проделанной над модулем оповещений, в главе 4 – описание работы над модулем статистики системы дистанционного обучения «МГУ Контест». Система реализована с использованием фреймворка для веб-приложений Django [6], написанного на языке программирования Python [7], в связи с этим расширение функциональности системы также производится с использованием указанных технологий.

Django использует шаблон проектирования Model-View-Controller – схему разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер, соответственно.

Интерактивные элементы интерфейса пользователя реализованы на языке программирования JavaScript [8]. В модуле статистики для реализации графического отображения собранных сведений используется инструмент Chart.js [9].

3 Модуль оповещения

3.1 Подписка на оповещения

Персонализация оповещений о событиях на сайте требует реализации механизма подписки пользователей на определенные сущности, о состоянии которых пользователь желает получать оповещения. Поэтому для предоставления преподавателям возможности получать оповещения о новых комментариях и посылках лишь к определенным курсам и разделам была реализована подписка пользователя на курсы и разделы. Реализация модели для сущности «Подписка» приведена в приложении (1).

На детальной странице каждого курса реализован интерфейс, доступный преподавателям, предоставляющий возможность подписаться на данный курс или отписаться от курса, а также подписаться на данный курс и все его разделы или отписаться от них. На детальной странице каждого раздела реализован интерфейс, доступный преподавателям, позволяющий подписаться на данный раздел или отписаться от него.

Все пользователи имеют возможность подписаться на оповещения об объявлениях и о расписании. Такую возможность предоставляет интерфейс настройки оповещений, реализация которого описана ниже.

3.2 Оповещения о событиях на сайте

Для пользователей сайта реализована возможность получать оповещения о следующих событиях:

- о добавлении комментария к курсу, разделу, задаче или посылке
- оповещения получают преподаватели, подписанные на курс или раздел, к которым написан данный комментарий, если же комментарий написан к задаче или посылке к задаче, то оповещения получают преподаватели, подписанные на раздел, включающий данную

задачу; реализация оповещения о комментариях приведена в приложении (2);

- о новой посылке к задаче – оповещения получают преподаватели, подписанные на раздел, включающий задачу, к которой отправлена посылка;
- о публикации объявления – оповещения получают все пользователи, находящиеся в группе, которой адресовано объявление, и подписанные на оповещения об объявлениях;
- о добавлении расписания – оповещения получают все пользователи, подписанные на оповещения о расписании;
- о создании вопроса – оповещение получает преподаватель, которому адресован вопрос, при переадресации вопроса или при появлении ответа на вопрос, оповещение об этом получает пользователь, задавший данный вопрос;
- о сообщении об ошибке – оповещения получают администраторы сайта, при изменении статуса сообщения оповещение получает пользователь, отправивший сообщение об ошибке.

3.3 Список оповещений

3.3.1 Бесконечная прокрутка списка оповещений

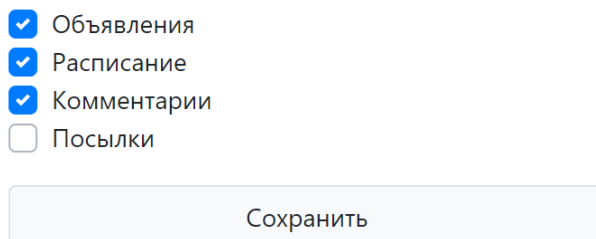
Для списка оповещений пользователя реализован механизм динамического раскрытия списка при прокручивании. При переходе на страницу списка оповещений отображаются лишь последние 30 полученных пользователем оповещений, при прокручивании страницы к концу отображаемого списка подгружаются следующие 30 более давних оповещений.

3.3.2 Динамическая отметка оповещений как прочтенных

В списке оповещений еще не прочтенные пользователем оповещения помечаются специальным цветом. При их просмотре в списке, то есть при нахождении таких оповещений в области видимости пользователя, они динамически помечаются как прочтенные, а также динамически обновляется значение счетчика непрочтенных оповещений. При просмотре объектов непрочтенных оповещений, например, комментария или объявления, о добавлении которых были получены данные оповещения, данные оповещения также помечаются как прочтенные. Реализация динамической отметки оповещений как прочтенных приведена в приложении (3).

3.4 Интерфейс настройки оповещений

3.4.1 Выбор типов получаемых оповещений



☒ Объявления
☒ Расписание
☒ Комментарии
☐ Посылки

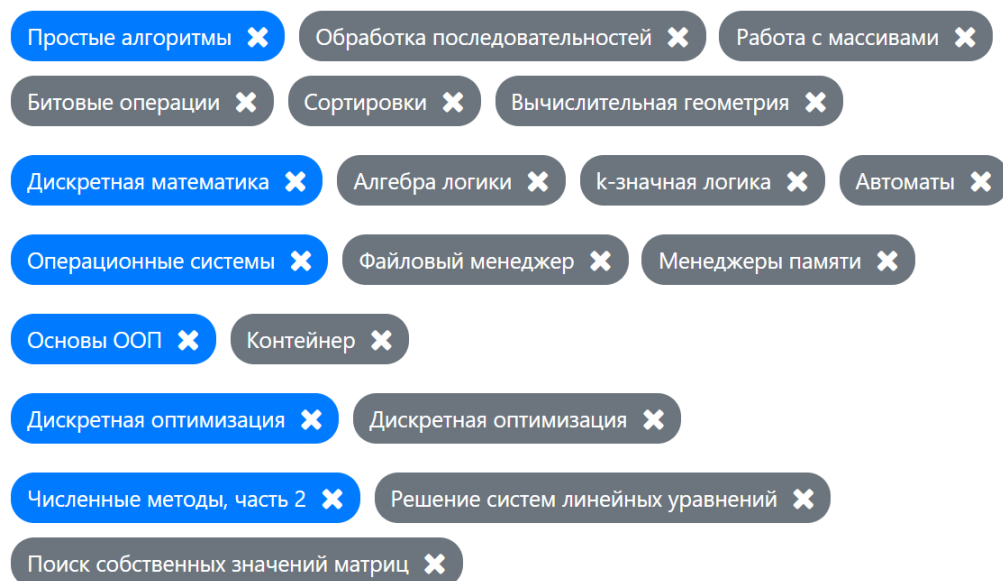
Сохранить

Пользователям предоставляется возможность выбора типов получаемых оповещений. Для этого реализован интерфейс настройки оповещений, где пользователь может отметить некоторые из следующих пунктов:

- объявления – доступен всем пользователям;
- расписание – доступен всем пользователям;
- комментарии – доступен преподавателям;
- посылки – доступен преподавателям.

При подписке на оповещения о комментариях или посылках при отсутствии у пользователя подписок на курсы или разделы, пользователь подписывается на все курсы и разделы. При отписке от оповещений о комментариях и посылках пользователь, соответственно, отписывается от всех курсов и разделов. Реализация представления интерфейса настройки оповещений приведена в приложении (4).

3.4.2 Список подписок на курсы и разделы



Интерфейс настройки оповещений включает также интерактивный список курсов и разделов, на которые подписан пользователь. Данный раздел интерфейса настройки оповещений реализован так, что по нажатию на элемент списка (курс или раздел) пользователь имеет возможность отписаться от выбранного курса или раздела. Список доступен преподавателям.

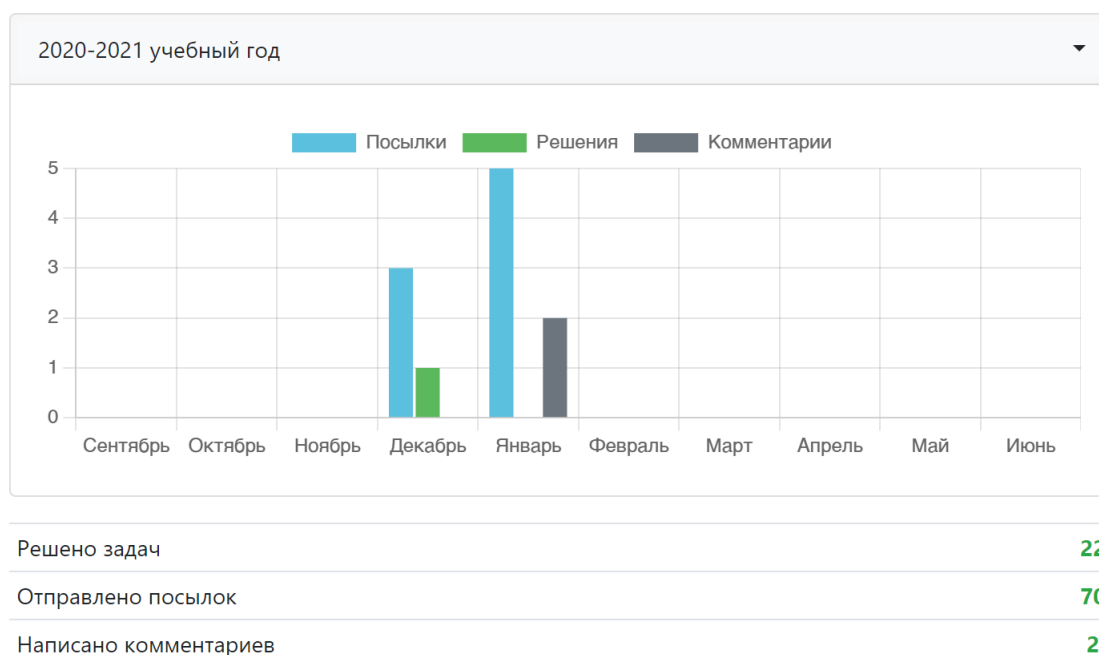
4 Модуль статистики

4.1 Профиль студента

4.1.1 Персонализированные изображения профиля

С целью усовершенствования отображения профиля пользователя реализована генерация персонализированных изображений профиля с инициалами имени и фамилии пользователя. Полученные изображения также отображаются в списке комментариев перед именем пользователя.

4.1.2 Сведения об активности



Реализовано отображение сведений об активности студента в виде столбчатой диаграммы, где по месяцам указаны количество посылок, сделанных студентом, количество решенных задач и количество комментариев. Имеется возможность выбрать для просмотра определенный учебный год или все время. Реализация диаграммы активности студента с помощью Chart.js приведена в приложении (5).

Также в сведениях об активности студента отображается общее количество решенных задач, посылок и комментариев студента.

4.1.3 Результаты курсов

Реализовано отображение результатов курсов, пройденных студентом, в профиле студента. Для каждого курса отображается уровень, ссылка на страницу с детальными результатами курса и оценка по данному курсу.

На странице с детальными результатами курса отображаются количества посылок к каждой из задач курса, оценки за каждую задачу курса, а также результаты решений задач, не входящих в обязательное задание студента по данному курсу, а решенных студентом дополнительно. Также здесь отображается в процентах в виде шкалы прогресса точность студента по данному курсу, подробности расчета которой описаны ниже.

4.1.4 Подсчет общей успеваемости и точности студента

Подсчет общей успеваемости студентов реализован на основании оценок студентов по пройденным курсам. Общая успеваемость отображается в процентах в виде шкалы прогресса в профиле студента.

Подсчет точности студента реализован на основании количества посылок к задаче до первой успешной, то есть до посылки, считающейся решением задачи. Пронумеруем все решенные студентом задачи от 1 до n , где n – общее количество решенных студентом задач. Тогда для каждой решенной студентом задачи с номером i точность A_i рассчитывается следующим образом:

$$A_i = \frac{100\%}{\lceil \frac{C_i}{L_i} \rceil},$$

где C_i – количество посылок к данной задаче до успешной, L_i – разрешенное количество посылок к данной задаче. Точность A студента является средней точностью этого студента по каждой из решенных задач, то есть

$$A = \frac{1}{n} \sum_{i=1}^n A_i.$$

Точность студента по отдельному курсу рассчитывается аналогично

на основании количества посылок до первой успешной к каждой из задач данного курса.

Точность профиля студента отображается в процентах в виде шкалы прогресса.

Реализация метода подсчета точности студента приведена в приложении (6)

4.2 Рейтинг студентов

На основании общей успеваемости студентов реализовано отображение рейтинга студентов указанного уровня, всех студентов, выпускников и отчисленных студентов. На основании успеваемости студентов по отдельным курсам реализовано отображение рейтинга студентов указанного уровня, всех студентов, выпускников и отчисленных студентов по данным курсам.

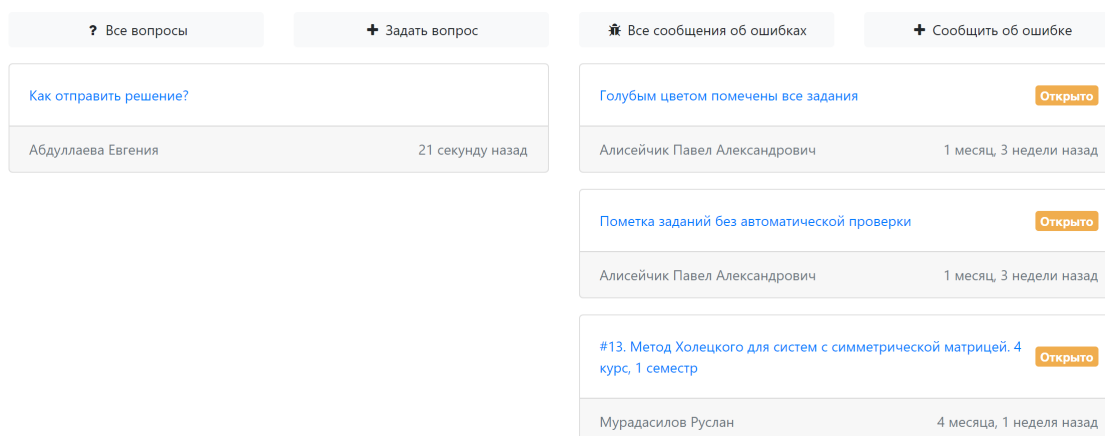
Отображение рейтинга студентов по общей успеваемости доступно всем преподавателям, по успеваемости по отдельным курсам – доступно преподавателям, ведущим соответствующие курсы.

4.3 Сведения о сложности курсов

Название	Средняя оценка	Сложность
Простые алгоритмы	3,9	Средний
Дискретная математика	4,1	Средний
Операционные системы	4,3	Средний
Основы ООП	4,4	Средний
Системы Программирования	0	Нет сведений
Дискретная оптимизация	4,2	Средний
Численные методы, часть 1	3,6	Средний
Численные методы, часть 2	3,4	Сложный

Для преподавателей в списке курсов, которые преподаватели ведут, реализовано отображение сведений о сложности данных курсов. Сложность курса определяется на основании средней оценки студентов, завершивших данный курс. В случае, если средняя оценка студентов по пятибалльной шкале составляет не менее 4.5 баллов, курс считается легким; иначе, в случае средней оценки студентов не менее 3.5 баллов, курс считается средним по сложности; иначе, в случае средней оценки студентов не менее 2-х баллов, курс считается сложным. Также в списке курсов отображается и средняя оценка студентов по каждому курсу.

5 Раздел помощи



В разделе помощи модели вопроса добавлено необязательное поле адресата для предоставления пользователям возможности задать вопрос конкретному преподавателю, соответственно переработана форма создания вопроса.

Реализована основная страница раздела помощи пользователям. На рассматриваемой странице для студента в виде карточек отображается список из пяти наиболее новых опубликованных или заданных данным студентом вопросов и список сообщений об ошибках со статусом «открыто», отправленных данным студентом; также на этой странице присутствуют ссылки в виде кнопок, позволяющие задать вопрос, отправить сообщение об ошибке, перейти к списку всех опубликованных или заданных этим студентом вопросов или списку всех отправленных студентом сообщений об ошибках. Для преподавателя и администратора на странице в виде карточек отображается список вопросов, у которых пока отсутствует ответ и список сообщений об ошибках со статусом «открыто»; также на этой странице присутствуют ссылки в виде кнопок, позволяющие задать вопрос, отправить сообщение об ошибке, перейти к списку всех вопросов или списку всех сообщений об ошибках.

Реализация представления раздела помощи пользователям приведена в приложении (7).

6 Заключение

В рамках настоящей выпускной квалификационной работы для системы дистанционного обучения «МГУ Контест» был реализован модуль оповещения, позволяющий пользователям персонализировать получаемые оповещения, и модуль статистики, открывающий преподавателям возможность наблюдать за детальной информацией об активности студентов, также внесены усовершенствования раздела помощи пользователям системы. Таким образом была расширена функциональность системы дистанционного обучения «МГУ Контест».

7 Список использованных источников и литературы

- [1] Моделирование процесса обучения / В. Б. Кудрявцев, П. А. Алисейчик, К. Вашик, Ж. Кнап, А. С. Строгалов, С. Г. Шеховцов // Интеллектуальные системы. – 2006, т. 10, вып. 1-4, стр. 189-270.
- [2] О дистанционном образовании – пример реализации и перспективы / П. А. Алисейчик, А. С. Строгалов, Р. А. Бекташев // Интеллектуальные системы. Теория и приложения. – 2016, т. 20, вып. 3, стр. 127-133.
- [3] Сайт «МГУ Контест»
<https://contest.msu-dev.ru>
- [4] Репозиторий проекта «МГУ Контест»
<https://github.com/ruslanbektashev/contest>
- [5] Изменения, внесенные в код проекта «МГУ Контест» в рамках данной выпускной квалификационной работы
<https://github.com/ruslanbektashev/contest/commits?author=eugeuie>
- [6] Документация фреймворка Django
<https://docs.djangoproject.com/en/3.0>
- [7] Документация языка Python
<https://docs.python.org/3>
- [8] Документация языка JavaScript
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [9] Документация инструмента Chart.js
<https://www.chartjs.org/docs/latest>

8 Приложения

В данном разделе приведены реализации некоторых ключевых усовершенствований, внедренных в систему «МГУ Контест» в рамках настоящей работы. Полный список изменений, внесенных в код [4] проекта «МГУ Контест» в рамках данной работы можно увидеть в репозитории проекта по ссылке [5].

1. Реализация модели для сущности «Подписка»:

```
class Subscription(models.Model):
    object_type = models.ForeignKey(ContentType,
                                    on_delete=models.CASCADE)
    object_id = models.PositiveIntegerField(null=True,
                                           blank=True)
    object = GenericForeignKey(ct_field='object_type')

    user = models.ForeignKey(User, on_delete=models.CASCADE)

    objects = SubscriptionManager
        .from_queryset(SubscriptionQuerySet)()

    class Meta:
        verbose_name = "Подписка"
        verbose_name_plural = "Подписки"
        unique_together = ('user', 'object_type', 'object_id')

    def validate_unique(self, exclude=None):
        if Subscription.objects.exclude(id=self.id).filter(
            user=self.user,
            object_type=self.object_type,
            object_id__isnull=True).exists():
```

```

        raise ValidationError("Подписка с такими
                                значениями полей User, Object type
                                и Object id уже существует.")
    super().validate_unique(exclude)

def __str__(self):
    return '%s, %s%s' % (self.user.account.get_full_name(),
                          self.object_type.model, ': ' + self.object.title
                          if self.object_id is not None else '')

```

2. Реализация метода для оповещения пользователей о новых комментариях. Данный метод вызывается при сохранении комментария, является методом модели Comment, соответствующей сущности «Комментарий».

```

def _notify_users(self, created=False):
    comment_subscribers_ids = Subscription.objects.filter(
        object_type=ContentType.objects.get(
            model='comment'
        )
    ).values_list('user', flat=True)
    parent_object_subscribers_ids =
        self.parent_object.subscription_set.values_list(
            'user_id',
            flat=True)
    user_ids = list(set(comment_subscribers_ids) &
                    set(parent_object_subscribers_ids))
    if self.object.owner_id not in user_ids:
        user_ids_set = set(user_ids)
        user_ids_set.add(self.object.owner_id)
        user_ids = list(user_ids_set)
    action = "оставил комментарий" if created
        else "изменил комментарий"

```

```

if self.id != self.parent_id:
    user = Comment.objects.get(id=self.parent_id).author
    Activity.objects.notify_user(
        user,
        subject=self.author,
        action="ответил на ваш комментарий",
        object=self,
        reference=self.object)
    if user.id in user_ids:
        user_ids.remove(user.id)
Activity.objects.notify_users(
    user_ids,
    subject=self.author,
    action=action,
    object=self,
    reference=self.object)

```

3. Реализация динамической отметки оповещений как прочтенных при просмотре:

```

class ReadWatcher {
    constructor(marker_url) {
        this.mark_activities_as_read_url = marker_url;
        this.activities =
            document.getElementById('activities');
        this.unread_activities_ids = [];
        this.options = {method: 'POST',
            headers: { 'ContentType': 'application/json' },
            cache: 'no-store' }
    }
}

```

```

updateUnreadCounterBadge() {
    let unread_activities_count_element = document
        .getElementById('unread_activities_count');
    let updated_count = parseInt(
        unread_activities_count_element.innerText) -
        this.unread_activities_ids.length;
    if (updated_count > 0)
        unread_activities_count_element.innerText =
            updated_count.toString();
    else
        unread_activities_count_element.remove();
}

```

```

markActivitiesAsRead() {
    if (this.unread_activities_ids.length > 0) {
        this.options['body'] =
            JSON.stringify({unread_activities_ids:
                this.unread_activities_ids });
        this.updateUnreadCounterBadge();
        this.unread_activities_ids = [];
        fetch(this.mark_activities_as_read_url,
            this.options).then(response => {});
    }
    setTimeout(this.markActivitiesAsRead.bind(this),
        1000);
}

```

```

isInViewPoint(element) {
    let bounding = element.getBoundingClientRect();
    return (bounding.top >= 0 && bounding.left >= 0
        && bounding.bottom <= (window.innerHeight ||

```

```

        document.documentElement.clientHeight)
        && bounding.right <= (window.innerWidth ||
        document.documentElement.clientWidth));
    }

    handleEvent(event) {
        for(let activity of this.activities.children) {
            let is_unread = JSON.parse(
                activity.getAttribute('data-unread'));
            if (is_unread) {
                let activity_id = parseInt(
                    activity.getAttribute('data-id'));
                let activity_body =
                    activity.getElementsByClassName(
                        'activity-body')[0];
                if (this.isInViewPoint(activity_body) &&
                    !this.unread_activities_ids.includes(
                        activity_id)) {
                    activity.classList.remove('bg-light');
                    activity.removeAttribute('data-unread');
                    this.unread_activities_ids
                        .push(activity_id);
                }
            }
        }
    }
}

```

4. Реализация представления интерфейса настройки оповещений:

```
class ManageSubscriptions(LoginRequiredMixin, FormView):
```

```

form_class = ManageSubscriptionsForm
template_name =
    'accounts/activity/activity_settings.html'

def get_initial(self):
    initial = super().get_initial()
    initial['object_type'] = list(
        self.request.user.subscription_set
            .values_list('object_type', flat=True))
    return initial

def get_form_kwargs(self):
    kwargs = super().get_form_kwargs()
    kwargs['user'] = self.request.user
    return kwargs

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    course_ids =
        self.request.user.subscription_set.filter(
            object_type=ContentType
                .objects.get_for_model(Course))
            .values_list('object_id', flat=True)
    context['all_courses'] = Course.objects.all()
    context['courses'] =
        Course.objects.filter(id__in=course_ids)
    contest_ids =
        self.request.user.subscription_set.filter(
            object_type=ContentType
                .objects.get_for_model(Contest))
            .values_list('object_id', flat=True)

```

```

context['contests'] =
    Contest.objects.filter(id__in=contest_ids)
return context

def post(self, request, *args, **kwargs):
    form = self.get_form()
    if form.is_valid():
        user = self.request.user
        choices = form.fields['object_type'].choices
        data = form.cleaned_data['object_type']
        user.subscription_set.filter(
            object_type__in=choices)
            .exclude(object_type__in=data).delete()
        new_object_type_ids = set(map(int, data)) -
            set(user.subscription_set
                .values_list('object_type', flat=True))
        new_subscriptions =
            (Subscription(object_type=
                ContentType.objects.get(id=i),
                user=user) for i in new_object_type_ids)
        Subscription.objects
            .bulk_create(new_subscriptions)
        if not Subscription.objects.for_user_models(
            user, Comment, Submission).exists():
            Subscription.objects.for_user_models(
                user, Course, Contest).delete()
        elif not Subscription.objects.for_user_models(
            user, Course, Contest).exists():
            for course in Course.objects.all():
                Subscription.objects.make_subscription(
                    user, object=course, cascade=True)

```

```

        return self.form_valid(form)
    return self.form_invalid(form)

def get_success_url(self):
    return reverse('accounts:activity-settings')

```

5. Реализация диаграммы активности студента с помощью Chart.js:

```

const submissionsChartData = {
  labels:
    [{% for item
      in year_month_submissions_solutions_comments %}
      {% if year == 0 %} '{{ item.1 }}' '{{ item.0 }}'
      {% else %} '{{ item.1 }}' {% endif %},
    {% endfor %}],
  datasets: [{
    label: 'Посылки',
    backgroundColor: 'rgb(91, 192, 222)',
    borderColor: 'rgb(91, 192, 222)',
    data: [{% for item in
      year_month_submissions_solutions_comments %}
      {{ item.2 }}, {% endfor %}],
  }, {
    label: 'Решения',
    backgroundColor: 'rgb(92, 184, 92)',
    borderColor: 'rgb(92, 184, 92)',
    data: [{% for item in
      year_month_submissions_solutions_comments %}
      {{ item.3 }}, {% endfor %}],
  }, {
    label: 'Комментарии',

```



```

        backgroundColor: 'rgb(108, 117, 125)',
        borderColor: 'rgb(108, 117, 125)',
        data: [{% for item in
                    year_month_submissions_solutions_comments %}
                    {{ item.4 }}, {% endfor %}],
    ]
};

```

```

const submissionsChartConfig = {
    type: 'bar',
    data: submissionsChartData,
    options: {
        scales: {
            y: {
                ticks: {
                    stepSize: 1
                }
            }
        }
    }
};

```

```

var submissionsChart = new Chart(
    document.getElementById('submissionsChart'),
    submissionsChartConfig
);

```

6. Реализация метода подсчета точности студента. Данный метод является методом модели Account, соответствующей сущности «Аккаунт».

```

def course_submissions_score(self, course_id=None):

```

```

Submission = apps.get_model('contests', 'Submission')
Assignment = apps.get_model('contests', 'Assignment')
submissions = Submission.objects.filter(owner=self.user)
if course_id:
    submissions = submissions
        .filter(problem__contest__course=course_id)
problem_ids = submissions
    .values_list('problem', flat=True)
    .distinct().order_by()
submissions_scores = []
for problem_id in problem_ids:
    problem_submissions =
        Submission.objects.filter(owner=self.user,
            problem=problem_id)
    success_submissions =
        problem_submissions.filter(status='OK')
    if success_submissions.exists():
        first_success_submission = success_submissions
            .order_by('date_created')[0]
        submissions_count = problem_submissions
            .filter(date_created__lte=
                first_success_submission.date_created)
            .count()
    else:
        submissions_count =
            problem_submissions.count() + 1
    assignment = problem_submissions.first().assignment
    submission_limit = assignment.submission_limit
    if assignment
        else Assignment.DEFAULT_SUBMISSION_LIMIT
    submissions_score = 100 / math.ceil(

```

```

        submissions_count / submission_limit)
    submissions_scores.append(submissions_score)
return round(mean(submissions_scores))
    if submissions_scores else 0

```

7. Реализация представления раздела помощи пользователям:

```

class Support(LoginRequiredMixin, ListView):
    model = Question
    template_name = 'support/index.html'
    context_object_name = 'questions'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        reports = Report.objects.filter(closed=False)
        if self.request.user
            .has_perm('support.change_report')
        else Report.objects.filter(
            Q(owner=self.request.user) & Q(closed=False))
        context.update({
            'reports': sorted(reports,
                key=attrgetter('date_created'),
                reverse=True)
        })
        return context

    def get_queryset(self):
        questions = Question.objects.filter(answer='')
        if self.request.user
            .has_perm('support.change_question')
        else Question.objects.filter(

```

```
        Q(is_published=True) |  
        Q(owner=self.request.user))  
return questions
```