```python
"""The module that contains all controller classes that are commmon to criminal
cases (criminal includes traffic). """
import os
import pathlib
from docxtpl import DocxTemplate
from loguru import logger

from PyQt5.QtWidgets import QDialog
from PyQt5 import QtCore
from PyQt5.QtCore import QDate
from PyQt5.QtWidgets import QLabel, QPushButton, QMessageBox, QComboBox, QLineEdit
from PyQt5.QtSql import QSqlDatabase, QSqlQuery

from models.case_information import (
    CaseInformation,
    CriminalCharge,
    AmendOffenseDetails,
    LicenseSuspensionTerms,
    CommunityControlTerms,
    CommunityServiceTerms,
    OtherConditionsTerms,
)
from views.add_conditions_dialog_ui import Ui_AddConditionsDialog
from views.amend_offense_dialog_ui import Ui_AmendOffenseDialog
from resources.db.DatabaseCreation import create_offense_list, create_statute_list


PATH = str(pathlib.Path().absolute())
TEMPLATE_PATH = PATH + "\\resources\\Templates\\"
SAVE_PATH = PATH + "\\resources\\Saved\\"
DB_PATH = PATH + "\\resources\\db\\"
CHARGES_DATABASE = DB_PATH + "\\charges.sqlite"


class BaseCriminalDialog(QDialog):
    """This class is a base class to provide methods that are used by some or
    all class controllers that are used in the application. This class is never
    instantiated as its own dialog, but the init contains the setup for all
    inherited class controllers."""

    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)
        self.doc = None
        self.docname = None
        self.pay_date_dict = {
            "forthwith": 0,
            "within 30 days": 30,
            "within 60 days": 60,
            "within 90 days": 90,
        }

    def close_window(self):
        """Function connected to a button to close the window. Can be connected
        to any button press/click/release to close a window."""
        self.close()

    @logger.catch
    def add_charge_process(self, bool):
        """The order of functions that are called when the add_charge_Button is
        clicked(). The order is important to make sure the informaiton is
        updated before the charge is added and the data cleared from the fields.

        The bool is passed as an argument through clicked() but not used."""
        self.criminal_charge = CriminalCharge()
        self.add_charge()
        self.clear_charge_fields()

    @logger.catch
    def create_entry_process(self):
        """The order of functions that are called when the create_entry_Button is pressed()
        on the MinorMisdemeanorDialog. The order is important to make sure the information is
        updated before the entry is created."""
        self.update_case_information()
        self.create_entry()
        self.close_event()

    @logger.catch
    def create_entry(self):
        """The standard function used to create an entry when a create entry
        button is press/click/released."""
        self.doc = DocxTemplate(self.template.template_path)
        self.doc.render(self.case_information.get_case_information())
        self.set_document_name()
        self.doc.save(SAVE_PATH + self.docname)
        os.startfile(SAVE_PATH + self.docname)

    def set_document_name(self):
        """Sets document name based on the case number and name of the template
        must include '.docx' to make it a Word document."""
        self.docname = (
            self.case_information.case_number + "_" + self.template.template_name + ".docx"
        )

    def set_case_information_banner(self):
        """Sets the banner on a view of the interface. It modifies label
        widgets on the view to text that was entered."""
        self.defendant_name_label.setText(
            "State of Ohio v. {defendant_first_name} {defendant_last_name}".format(
                defendant_first_name = self.case_information.defendant.first_name,
                defendant_last_name = self.case_information.defendant.last_name
                )
            )
        self.case_number_label.setText(self.case_information.case_number)
        if self.case_information.defendant_attorney_name is not None:
            self.defendant_attorney_name_label.setText(
                "Attorney: " + self.case_information.defendant_attorney_name
            )
        else:
            self.defendant_attorney_name_label.setText("Attorney: None")

    def update_case_information(self):
        """Placeholder for future use to refactor out of other dialogs and reduce
        code reuse."""
        pass


class AddConditionsDialog(BaseCriminalDialog, Ui_AddConditionsDialog):
    """The AddConditionsDialog is created when the addConditionsButton is clicked on
    the MinorMisdemeanorDialog. The conditions that are available to enter information
    for are based on the checkboxes that are checked on the MMD screen."""

    @logger.catch
    def __init__(self, minor_misdemeanor_dialog, parent=None):
        super().__init__(parent)
        self.case_information = minor_misdemeanor_dialog.case_information
        self.modify_view()
        self.community_service = (
            minor_misdemeanor_dialog.community_service_checkBox.isChecked()
        )
        self.license_suspension = (
            minor_misdemeanor_dialog.license_suspension_checkBox.isChecked()
        )
        self.community_control = (
            minor_misdemeanor_dialog.community_control_checkBox.isChecked()
        )
        self.other_conditions = (
            minor_misdemeanor_dialog.other_conditions_checkBox.isChecked()
        )
        self.other_conditions = (
            minor_misdemeanor_dialog.other_conditions_checkBox.isChecked()
        )
        self.enable_condition_frames()

    @logger.catch
    def modify_view(self):
        """Modifies the view of AddConditionsDialog that is created by the UI
        file.
        Gets the total number of charges from the charges in charges_list then
        loops through the charges_list and adds parts of each charge to the
        view. CLEAN UP?"""
        index_of_charge_to_add = 0
        column = self.charges_gridLayout.columnCount() + 1
        total_charges_to_add = len(self.case_information.charges_list)
        while index_of_charge_to_add < total_charges_to_add:
            charge = vars(self.case_information.charges_list[index_of_charge_to_add])
            if charge is not None:
                self.charges_gridLayout.addWidget(
                    QLabel(charge.get("offense")), 0, column
                )
                self.charges_gridLayout.addWidget(
                    QLabel(charge.get("statute")), 1, column
                )
                self.charges_gridLayout.addWidget(
                    QLabel(charge.get("finding")), 2, column
                )
                column += 1
                index_of_charge_to_add += 1

    @logger.catch
    def enable_condition_frames(self):
        """Enables the frames on the AddConditionsDialog dialog if the condition is checked
        on the MinorMisdemeanorDialog screen. Also creates an instance of the object for
        each condition."""
        if self.other_conditions is True:
            self.other_conditions_frame.setEnabled(True)
            self.other_conditions_details = OtherConditionsTerms()
        if self.license_suspension is True:
            self.license_suspension_frame.setEnabled(True)
            self.license_suspension_details = LicenseSuspensionTerms()
            self.license_suspension_date_box.setDate(QtCore.QDate.currentDate())
        if self.community_service is True:
            self.community_service_frame.setEnabled(True)
            self.community_service_terms = CommunityServiceTerms()
            self.community_service_date_to_complete_box.setDate(
                QtCore.QDate.currentDate()
            )
        if self.community_control is True:
            self.community_control_frame.setEnabled(True)
            self.community_control_terms = CommunityControlTerms()

    @logger.catch
    def add_conditions(self):
        """The method is connected to the pressed() signal of continue_Button on the
        Add Conditions screen."""
        if self.community_service is True:
            self.add_community_service_terms()
        if self.community_control is True:
            self.add_community_control_terms()
        if self.license_suspension is True:
            self.add_license_suspension_details()
        if self.other_conditions is True:
            self.add_other_condition_details()

    @logger.catch
    def add_community_control_terms(self):
        """The method adds the data entered to the CommunityControlTerms object
        that is created when the dialog is initialized. Then the data is transferred
        to case_information."""
        self.community_control_terms.type_of_community_control = (
            self.type_of_community_control_box.currentText()
        )
        self.community_control_terms.term_of_community_control = (
            self.term_of_community_control_box.currentText()
        )
        self.case_information.community_control_terms = self.community_control_terms

    @logger.catch
    def add_community_service_terms(self):
        """The method adds the data entered to the CommunityServiceTerms object
        that is created when the dialog is initialized. Then the data is transferred
        to case_information."""
        self.community_service_terms.hours_of_service = (
            self.community_service_hours_ordered_box.value()
        )
        self.community_service_terms.days_to_complete_service = (
            self.community_service_days_to_complete_box.currentText()
        )
        self.community_service_terms.due_date_for_service = (
            self.community_service_date_to_complete_box.date().toString("MMMM dd, yyyy")
        )
        self.case_information.community_service_terms = self.community_service_terms

    @logger.catch
    def add_license_suspension_details(self):
        """The method adds the data entered to the LicenseSuspensionTerms object
        that is created when the dialog is initialized. Then the data is transferred
        to case_information."""
        self.license_suspension_details.license_type = (
            self.license_type_box.currentText()
        )
        self.license_suspension_details.license_suspended_date = (
            self.license_suspension_date_box.date().toString("MMMM dd, yyyy")
        )
        self.license_suspension_details.license_suspension_term = (
            self.term_of_suspension_box.currentText()
        )
        if self.remedial_driving_class_checkBox.isChecked():
            self.license_suspension_details.remedial_driving_class_required = True
        else:
            self.license_suspension_details.remedial_driving_class_required = False
        self.case_information.license_suspension_details = (
            self.license_suspension_details
        )

    @logger.catch
    def add_other_condition_details(self):
        """The method allows for adding other conditions based on free form text
        entry."""
        self.other_conditions_details.other_conditions_terms = (
            self.other_conditions_plainTextEdit.toPlainText()
        )
        self.case_information.other_conditions_details = (
            self.other_conditions_details
        )


    @logger.catch
    def set_service_date(self, days_to_complete):
        """Sets the community_service_date_to_complete_box based on the number
        of days chosen in the community_service_date_to_complete_box."""
        days_to_complete = int(
            self.community_service_days_to_complete_box.currentText()
        )
        self.community_service_date_to_complete_box.setDate(
            QDate.currentDate().addDays(days_to_complete)
        )


class AmendOffenseDialog(BaseCriminalDialog, Ui_AmendOffenseDialog):
    """The AddOffenseDialog is created when the amendOffenseButton is clicked on
    the MinorMisdemeanorDialog screen. The case information is passed in
    order to populate the case information banner.

    The set_case_information_banner is an inherited method from BaseCriminalDialog."""
    @logger.catch
    def __init__(self, case_information, parent=None):
        super().__init__(parent)
        self.case_information = case_information
        self.amend_offense_details = AmendOffenseDetails()
        self.set_case_information_banner()
        self.modify_view()

    @logger.catch
    def modify_view(self):
        """The modify view method updates the view that is created on init.
        Place items in this method that can't be added directly in QtDesigner
        so that they don't need to be changed in the view file each time pyuic5
        is run."""
        offense_list = create_offense_list()
        self.original_charge_box.addItems(offense_list)
        self.amended_charge_box.addItems(offense_list)

    @logger.catch
    def amend_offense(self):
        """Adds the data entered for the amended offense to the AmendOffenseDetails
        object then points the case_information object to the AmendOffenseDetails
        object."""
        self.amend_offense_details.original_charge = (
            self.original_charge_box.currentText()
        )
        self.amend_offense_details.amended_charge = (
            self.amended_charge_box.currentText()
        )
        self.amend_offense_details.motion_disposition = (
            self.motion_decision_box.currentText()
        )
        self.case_information.amend_offense_details = self.amend_offense_details
```